Universidad de las Ciencias Informáticas

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título:

Componente de medición de la calidad de imágenes de huellas dactilares.

Autores:

Miriela Velázquez Arias

Alexei Alayo Rondón

Tutores:

Ing. Ramón Santana Fernández

Ing. Yaicel Díaz Córdova

Co-tutor:

Msc. Martha Ambruster Crespo

La Habana, junio de 2014

Declaración de autoría

Declaramos ser los autores del trabajo de diploma titulado “Componente para la medición de la calidad de las imágenes de huellas dactilares” y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los ____ días del mes de _____ del año _____.

Autores:

_____ _____

Alexei Alayo Rondón Miriela Velázquez Arias

Tutores:

_____ _____

Ramón Santana Fernández Yaicel Díaz Córdova

Co-tutor:

_____

Martha Ambruster Crespo

Datos de contacto

Agradecimientos

Dedicatoria

Resumen

Los sistemas biométricos son sistemas automatizados para el reconocimiento de individuos que fundamentan sus decisiones en los rasgos conductuales o físicos intrínsecos de los mismos. De manera particular, los sistemas automáticos de identificación dactilar permiten la verificación o identificación de una persona a partir del análisis de sus impresiones dactilares.

La calidad de las imágenes de huellas dactilares, capturadas como parte del proceso de enrolamiento de usuarios en un sistema biométrico, influye directamente en la complejidad de los algoritmos de extracción que se requieran. Factores como el scanner utilizado para la captura, la presión ejercida sobre el mismo, y las condiciones de la piel, afectan la calidad de la imagen obtenida, por lo que resulta conveniente realizar un análisis del índice de calidad de dichas imágenes durante el proceso de enrolamiento, garantizando que se excluyan del proceso de extracción las imágenes de baja calidad.

La presente investigación tiene la finalidad de desarrollar un componente de medición de calidad de imágenes de huellas dactilares que pueda ser utilizado por cualquier sistema de enrolamiento biométrico. En el documento se recogen los resultados de la investigación realizada, describiendo herramientas, tecnologías utilizadas y artefactos generados como parte del proceso de desarrollo guiado por XP.

Palabras clave: huellas dactilares, medición, calidad, biometría.

Índice de Contenido

Índice de Figuras

Índice de Tablas

Introducción

En la actualidad los sistemas de reconocimiento biométrico están aportando otro nivel, en términos de seguridad, a los sistemas tradicionales de identificación personal, o sea, a los métodos basados en algo que la persona sabe (contraseña, PIN) o en algo que la persona tiene (tarjetas de identificación) (1).

Un sistema biométrico es esencialmente un sistema de reconocimiento de patrones que, a partir de la adquisición de los datos biométricos de un individuo, extrae un conjunto de características y las compara con las almacenadas previamente, emitiendo un nivel de similitud (2).

El término biometría se refriere al uso de características distintivas anatómicas o de comportamiento, denominadas rasgos o identificadores biométricos, para el reconocimiento automático de individuos (3). Entre las características físicas más utilizadas en la actualidad se encuentran las huellas dactilares, la retina, el iris, el rostro y la geometría de la palma de la mano. La firma manuscrita y el modo de caminar son algunas de las características dinámicas o de comportamiento.

El éxito de la tecnología basada en huellas dactilares en aplicaciones para el cumplimiento de la ley, la disminución del costo de dispositivos de detección de huellas dactilares, la creciente disponibilidad de recursos de computación de bajo costo, y el incremento de fraudes mediante el robo de identidad, han dado paso al aumento gradual del uso de este tipo de tecnología en dominios comerciales, gubernamentales, civiles y financieros (3).

Un Sistema Automático de Identificación Dactilar (SAID) es un conjunto de componentes físicos y lógicos que permiten la verificación o identificación de una persona a partir del análisis de sus impresiones dactilares. El SAID tiene tres modos de funcionamiento en el reconocimiento de individuos: enrolamiento[1], verificación[2] e identificación[3] (1).

El enrolamiento es el proceso mediante el cual se registran individuos en la base de datos del sistema biométrico. El módulo de captura, encargado de la adquisición de la imagen de la huella dactilar, es el primero que tiene lugar como parte de este proceso. Luego de la captura de la huella, se lleva a cabo el mejoramiento de la imagen para luego realizar la extracción de minucias de la huella dactilar. Esta secuencia de subprocesos permite conformar una plantilla de minucias, la cual es almacenada junto a otros datos del usuario, completando así el enrolamiento del mismo.

El módulo de captura digitaliza la imagen de la huella dactilar. Esta etapa es fundamental, ya que según la calidad de dicha imagen será la complejidad de los algoritmos de extracción que se requieran, lo que repercutirá en la eficiencia y rapidez del sistema (4).

---

[1] **Modo de enrolamiento**: En este modo se extraen las características de las huellas dactilares y se guardan para poder ser comparadas.

[2] **Modo de verificación**: En la que dos impresiones (una de persona conocida y otra desconocida) se cotejan a fin de conocer si la identidad de la segunda persona corresponde con la primera.

[3] **Modo de identificación**: En la que se dispone de una impresión de una persona desconocida y de una base de datos, más o menos voluminosa, de impresiones dactilares. La identificación de la persona se realiza mediante una búsqueda en dicha base de datos.

Resulta estratégico realizar un chequeo de calidad de las imágenes capturadas, de modo que se asegure que la muestra adquirida pueda ser procesada sin dificultades en etapas posteriores. La calidad de las imágenes se ve influenciada por diversos factores:

1. La captura de las imágenes se realiza a través de dispositivos (scanner) con sensores de huellas dactilares, los cuales abarcan un amplio abanico de posibilidades (sensores ópticos, térmicos, capacitivos, ultrasónicos). Debido a esta variedad de sensores, y a que la presión que ejerce un individuo sobre el mismo puede variar, diversas capturas de una huella no dan lugar a imágenes totalmente idénticas, por lo que dichas imágenes son generalmente examinadas antes de extraer las características correspondientes.
2. Independientemente del sensor empleado para su captura, en la imagen influye la condición de la piel. Por un lado la piel seca tiende a causar inconsistencia en el contacto de las crestas de la huella con la superficie del sensor, causando la ruptura de estas y la aparición de píxeles claros en su estructura, y por otro lado, en la piel húmeda los valles tienden a llenarse con sudor, causando que aparezcan zonas oscuras en la imagen. Ambas condiciones contribuyen al ruido, la degradación de la imagen y la detección de falsas minucias.

El proceso de extracción de características se ve afectado directamente por la calidad de la imagen de la huella dactilar. Si la imagen capturada es de mala calidad, el algoritmo de extracción de características puede extraer un conjunto de minucias que en realidad no existan, o no tengan valor identificativo, denominadas falsas minucias. Este error se conoce como *Falla al Procesar*, o FTP por sus siglas en inglés (3).

La Universidad de las Ciencias Informáticas (UCI) es una universidad científico-productiva que ha desarrollado soluciones en la esfera de la biometría, dentro de las que se pueden mencionar un sintetizador de huellas dactilares y componentes para la extracción de minucias, clasificación y comparación de huellas dactilares.

El Centro de Identificación y Seguridad Digital (CISED), de la UCI, realizó la implementación de un nuevo sistema de identificación y control de acceso para la universidad (SICA). Uno de los objetivos principales de este nuevo sistema es realizar el cambio de credencial identificativa al personal autorizado de la universidad. Entre los requisitos del sistema se encuentran la captura, procesamiento y almacenamiento de las impresiones dactilares de cada uno de sus trabajadores y estudiantes. Para ello se desarrolló un componente de captura de datos utilizando el módulo de extracción del SOURCEAFIS.

Durante el despliegue del SICA fue detectado que varias impresiones, debido a diferentes factores, estaban ingresando al sistema imágenes de baja calidad, afectando el funcionamiento del componente de extracción. Para solucionar el problema se realizó la medición de la calidad identificativa de la huella dactilar, proceso que impacta en el rendimiento del sistema debido a que se ejecuta una extracción de características en el momento de la captura, se realiza un conteo de la cantidad de minucias obtenidas y se determina si la plantilla puede ser almacenada. Sin embargo este proceso no garantiza la calidad con la que son extraídas las características identificativas.

En el Departamento de Componentes del CISED se desarrolla actualmente, en la línea de biometría, un componente de extracción de minucias, el cual está dirigido a realizar el pre-procesamiento de una imagen de huella dactilar para luego ejecutar la extracción de minucias de la misma. Debido a que no se cuenta con un mecanismo que determine la calidad de dicha imagen, se corre el riesgo de enfrentar situaciones negativas similares a lo acontecido durante el despliegue del SICA.

Teniendo en cuenta que una alta tasa del error FTP incide directamente en la eficiencia de un sistema biométrico, y que en la calidad de las imágenes de huellas dactilares influyen tanto el sensor como la presión ejercida por el dedo y las condiciones de la piel, las imágenes capturadas deben ser sometidas a un análisis de calidad, con el fin de descartar aquellas donde con mayor probabilidad pueda fallar el algoritmo de extracción en la misión de extraer un conjunto utilizable de características. Por esta razón se hace indispensable el desarrollo de estrategias encaminadas a reducir dicha tasa de error y a garantizar que no se registren minucias en zonas de baja calidad de la imagen, incrementando así la calidad y confiabilidad del sistema biométrico.

De la situación problemática anterior se identifica como **problema de la investigación:** ¿Cómo determinar la calidad de las imágenes de huellas dactilares para decidir su ingreso en el módulo de extracción?

Teniendo en cuenta el problema definido con anterioridad se determina como **objeto de estudio** los procesos de medición de calidad de imágenes de huellas dactilares.

Como **objetivo general** se plantea desarrollar un componente que permita determinar la calidad de la imagen de una huella dactilar y que posibilite descartar imágenes de baja calidad del proceso de extracción de minucias para ser utilizado por el componente de extracción de minucias que se desarrollará en el Departamento de Componentes del CISED, derivando en los siguientes **objetivos específicos**:

- Determinar las tendencias mundiales de los algoritmos de medición de calidad de imágenes de huellas dactilares.
- Definir las métricas por las que se medirá la calidad de las imágenes de huellas dactilares.
- Definir los algoritmos de medición de calidad de imágenes de huellas dactilares a utilizar.
- Analizar las tecnologías, metodologías y herramientas existentes que contribuyan al desarrollo del componente.
- Desarrollar el proceso de medición de calidad de imágenes de huellas dactilares.
- Realizar pruebas al componente.

Para cumplimentar los objetivos anteriores, se plantean como **tareas de la investigación** las siguientes:

1) Definición de aspectos teóricos asociados al proceso de medición de calidad de imágenes de huellas dactilares.
2) Análisis del estado del arte en Cuba y el mundo referente a la medición de calidad de imágenes de huellas dactilares.
3) Definición de la metodología, las herramientas y tecnologías que soporten el componente.
4) Especificación de requisitos funcionales y no funcionales del software.

5) Definición de la arquitectura de software del componente.
6) Definición de los patrones de diseño.
7) Implementación de los algoritmos de medición de calidad de imágenes de huellas dactilares.
8) Realización de pruebas unitarias y de aceptación al componente.

Se propone como **idea a defender** que la determinación de la calidad de la imagen de una huella dactilar antes de realizar el proceso de extracción permite seleccionar imágenes con mejores condiciones para obtener buenos resultados en la comparación de características.

Los **Métodos Científicos** utilizados son:

Métodos teóricos:

- **Analítico-Sintético:** se hace uso de este método para analizar los elementos teóricos asociados al proceso de medición de calidad de las imágenes de huellas dactilares y su calidad, así como para examinar los algoritmos de medición de calidad de estas imágenes.

- **Análisis Histórico-Lógico:** facilita el análisis del desarrollo de los algoritmos de medición de la calidad de imágenes de huellas dactilares de manera cronológica.

- **Modelación:** la utilización de este método permite la confección de los diagramas correspondientes al diseño del software, con el fin de facilitar la comprensión del funcionamiento del componente.

Métodos empíricos:

- **Experimentación:** se emplea este método con el propósito de comprobar las funcionalidades según son implementadas.

- **Entrevista:** se hace uso de este método para interactuar con personal capacitado, con el fin de obtener información e ideas que contribuyan al desarrollo de la investigación.

El presente trabajo de diploma consta de tres capítulos estructurados de la siguiente forma:

- **Capítulo I Fundamentación teórica:** se realiza el estudio del estado del arte de algoritmos de medición de calidad de imágenes de huellas dactilares, se abordan elementos teóricos vinculados a la investigación, y se analizan metodologías de desarrollo, tecnologías y herramientas a utilizar en el desarrollo del componente.
- **Capítulo II: Características del componente:** se describe la solución propuesta y se ofrecen detalles de los principales aspectos relacionados con su diseño. Se explica la dinámica del componente a través de las historias de usuario y otros modelos auxiliares.
- **Capítulo III: Implementación y prueba:** en este capítulo se describen los artefactos relacionados con la implementación y las pruebas realizadas al componente, con el objetivo de validar su correcto funcionamiento y su correspondencia con los requerimientos especificados previamente.

Justificación de la investigación

En un sistema de reconocimiento de personas mediante huellas dactilares, el proceso de extracción de características es crucial para obtener buenos resultados medidos en tasas de falsos aceptados y falsos rechazos durante el proceso de comparación. La extracción de características recibe como entrada una imagen de huella dactilar y como salida del proceso brinda una plantilla de minucias, la cual representa la información identificativa del individuo al que pertenece. Factores como el ruido, pequeñas partículas consideradas basura, la nitidez de la imagen, los factores morfológicos de la huella dactilar, entre otros, pueden afectar el patrón de crestas y la imagen dactilar en general, trayendo como consecuencia el análisis de imágenes de baja calidad que arrojan resultados con poco valor identificativo, afectando los procesos de extracción y comparación de minucias.

Capítulo I: Fundamentación Teórica

Introducción

En el presente capítulo se exponen los principales conceptos asociados al dominio del problema y se realiza un estudio del estado del arte a nivel nacional e internacional sobre algoritmos y componentes de medición de huellas dactilares. Se analizan además las metodologías, tecnologías y herramientas a utilizar para el desarrollo de la investigación.

1.1 Huellas dactilares

Las huellas dactilares son patrones constituidos por las crestas papilares de los dedos de las manos que aparecen visibles en la epidermis. Se forman en el periodo fetal a partir del sexto mes de vida y se mantienen invariantes durante toda la vida del individuo. Alteraciones como quemaduras o accidentes que afecten la epidermis de los dedos son factores que podrían afectar las huellas dactilares (5).

1.1.1 Propiedades

1. **Perennidad**: las crestas se mantienen en la piel durante toda la vida (salvo traumatismos graves). La única diferencia en el dibujo papilar de una persona durante su crecimiento es el cambio de tamaño y anchura de las crestas, pero no su disposición espacial ni en su tipología.
2. **Inmutabilidad:** las crestas son invariables en número, forma, situación y dirección. Ninguna enfermedad modifica el patrón.
3. **Diversidad:** los patrones son distintos para cada persona y para cada dedo de una misma persona. Incluso son distintos los dibujos entre gemelos univitelinos (1).

1.1.2 Características estructurales

Las huellas dactilares están formadas por:

- **Crestas y Valles: l**as huellas dactilares están constituidas por rugosidades que forman salientes y depresiones. Las salientes se denominan crestas papilares y las depresiones surcos inter-papilares o valles (6).

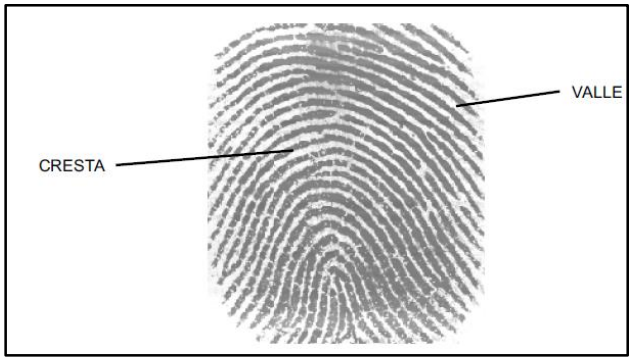


Figura 1 Crestas y Valles de una huella dactilar.

Las características identificativas de una huella dactilar están definidas por las minucias, las cuales no son más que pequeñas discontinuidades formadas por el cruce y terminación de las crestas (7).

- **Minucia**: punto de interés de la huella dactilar que se representa como: $\boldsymbol{minucia} = \{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}\}$ donde $\boldsymbol{x}$ y $\boldsymbol{y}$ son la posición en la imagen de la huella, y $\boldsymbol{\theta}$ es el ángulo de la minucia. Abarcan diversas clasificaciones (ver figura 2). Las clases de mayor valor significativo son las terminaciones y bifurcaciones, el resto puede verse como la combinación de estas.

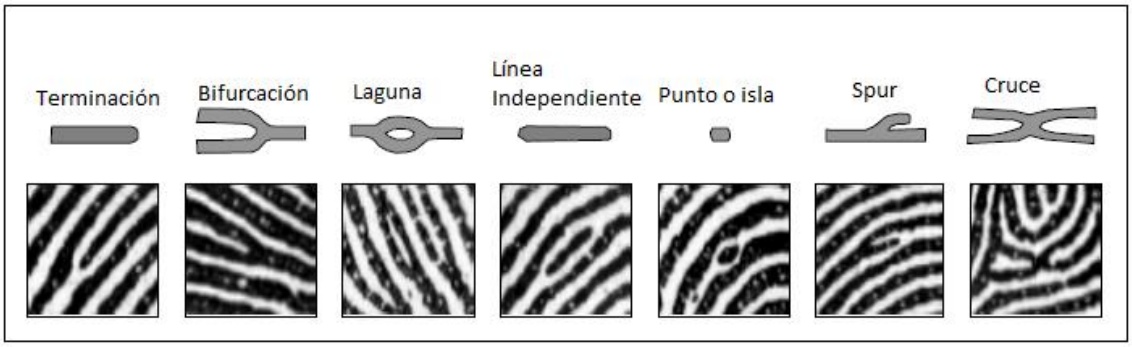
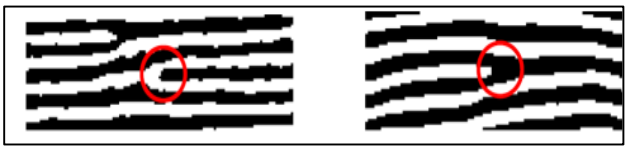


Figura 2 Tipos de minucias.

  - **Terminación**: una minucia de terminación es donde la cresta termina abruptamente.
  - **Bifurcación:** una minucia de bifurcación es donde la cresta se divide (8).



a) **b)**

Figura 3 a) Minucia de terminación, b) Minucia de bifurcación.

- **Core:** es el punto de máxima curvatura de la cresta.
- **Delta:** es el centro de un conjunto de crestas inferiores al núcleo que dibuja varios triángulos concéntricos (7).

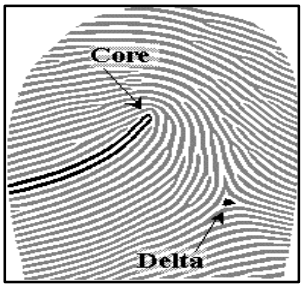


Figura 4 Core y Delta.

1.2 Calidad de Imágenes de Huellas Dactilares

La calidad de las imágenes de huellas dactilares se define como una medida de la claridad de las crestas y valles y la capacidad de extracción de las características utilizadas para la identificación (9). La mayor parte de los esquemas operacionales de huellas digitales para la estimación de calidad de imagen de huellas dactilares se centran en la utilidad de las imágenes (10).

Estimar la calidad de las imágenes de las huellas dactilares permite (3):

- Rechazar las muestras de muy baja calidad durante el enrolamiento y/o para seleccionar la mejor muestra(s).
- Aislar regiones irrecuperables, donde mejora la huella digital es contraproducente, ya que conduce a la detección de varias características espurias[4].
- Seleccionar entre diferentes estrategias de extracción de minucias.
- Asignar pesos a las características (en la fase de comparación) de acuerdo a la calidad.

1.3 Estado del arte

1.3.1 Sistemas y componentes que realizan el proceso de análisis de calidad

Existen múltiples soluciones biométricas que implementan mecanismos de medición de calidad de imágenes de huellas dactilares.

- **Marco internacional**

  - **NBIS**

La distribución NIST Biometric Image Software (NBIS) es desarrollada por el Instituto Nacional de Estándares y Tecnología (NIST) del Buró Federal de Investigaciones (FBI) y el Departamento de Seguridad Nacional (DHS). Entre sus características se encuentra el desarrollo de una nueva revisión del algoritmo NFIQ (*NIST Fingerprint Image Quality*), un algoritmo de medición de calidad de imágenes de huellas dactilares usado para predecir el rendimiento de los algoritmos de comparación (11).

El algoritmo NFIQ analiza la imagen de la huella y le asigna un valor de calidad de 1 (máxima calidad) a 5 (calidad más baja) a la imagen. Las imágenes de mayor calidad producen un rendimiento significativamente mejor con algoritmos de comparación. La distribución NBIS contiene código fuente implementado en C para el procesamiento y análisis de datos biométricos.

  - **QualityCheck**

QualityCheck es una biblioteca para la medición de la calidad de las imágenes de huellas dactilares que forma parte de los SDK[5] *SequenceCheck* y *Aware WSQ1000*. Utiliza algoritmos avanzados para evaluar si la imagen

[4] **Característica espuria:** punto que se forma como resultado del procesamiento de las imágenes de huellas dactilares. Es un punto sin valor identificativo, ya que no forma parte de la huella dactilar. También conocida como falsa minucia.

[5] **SDK:** kit de desarrollo de software (Software Development Kit por sus siglas en inglés).

de la huella dactilar tiene una calidad aceptable para la comparación biométrica. Implementa una medición específica de la calidad de la imagen dactilar basada en la continuidad del flujo de crestas en todas las regiones de la imagen dactilar. QualityCheck genera una calificación general que oscila entre 0 y 100 y suministra información sobre las áreas de la imagen que presentan problemas. Dicha información se ofrece codificada por colores (ver figura 5) (12).

QualityCheck define sus umbrales típicos de calidad de la siguiente forma:

- 85-100 Buena.
- 75-84 Adecuada.
- 60-74 Marginal.
- 0-59 Mala.

Azul	áreas borrosas o quebradas
Rojo	áreas muy oscuras
Amarillo	áreas muy claras
Verde	áreas de buena calidad

Figura 5 Codificación por colores de regiones de la huella dactilar del QualityCheck.

- **AccuScan**

AccuScan es un módulo adicional de NISTPack[6] que representa una solución certificada por el FBI para escanear y digitalizar fichas decadactilares[7] impresas en papel. Esta herramienta, junto al NISTPack, permite (13):

- Cumplimiento de las especificaciones para escáneres del Apéndice F del FBI IQS[8].
- Compresión y descompresión WSQ con certificación del FBI.
- Medición y aseguramiento de la calidad de las imágenes de huellas dactilares.
- Lectura, escritura, visualización y edición de transacciones decadactilares de postulantes a la certificación del FBI IQS.

➢ **Marco nacional**

- **Componente para determinar la calidad en imágenes de huellas dactilares de la UCI**

[6] **NISTPack:** SDK que habilita aplicaciones para visualizar, editar y validar transacciones de datos biométricos en cumplimiento de las normas ANSI/NIST-ITL.

[7] **Ficha decadactilar:** ficha personal que incluye la impresión de las huellas dactilares de cada uno de los diez dedos de las manos. Registro de identificación dactiloscópica.

[8] **FBI IQS:** Especificaciones de Calidad de Imágenes del FBI (*FBI Image Quality Specification*).

El componente desarrollado por miembros de los centros CISED y GEYSED (Centro de Desarrollo de Geoinformática y Señales Digitales) de la UCI, implementa un método para evaluar la calidad de las imágenes dactilares de manera local. Para ello se emplea una estrategia para estimar la calidad de la huella dactilar utilizando la media, varianza, desviación estándar, suavidad, uniformidad, homogeneidad y contraste direccional. Dependiendo de la cantidad total de bloques de la imagen en buen o mal estado, se clasifica la imagen desde 1(mayor calidad) hasta 5 (peor calidad) (14).

- **Biomesys AFIS**

Biomesys AFIS es un SAID desarrollado por DATYS, Tecnologías y Sistemas desde el 2006. Está diseñado para reducir los tiempos y aumentar la efectividad, en la identificación y autenticación de personas, a partir de las impresiones dactilares (15). El sistema, en su versión civil, es capaz de registrar y almacenar millones de fichas decadactilares (16). La actualización de cada ficha decadactilar se hace teniendo en cuenta el nivel de calidad de las imágenes y realizando reportes a la administración del sistema cada vez que se produzca una actualización. En la medición de la calidad de las impresiones dactilares se emplea una escala de 1 a 5 según las normas internacionales. Para facilitar la interpretación por parte de los usuarios, se vincula esta escala numérica con colores como se muestra en la figura 6.

- Buena Calidad, 1 y 2, color Azul.
- Mediana Calidad, 3, color Amarillo.
- Mala Calidad, 4 y 5, color Rojo.

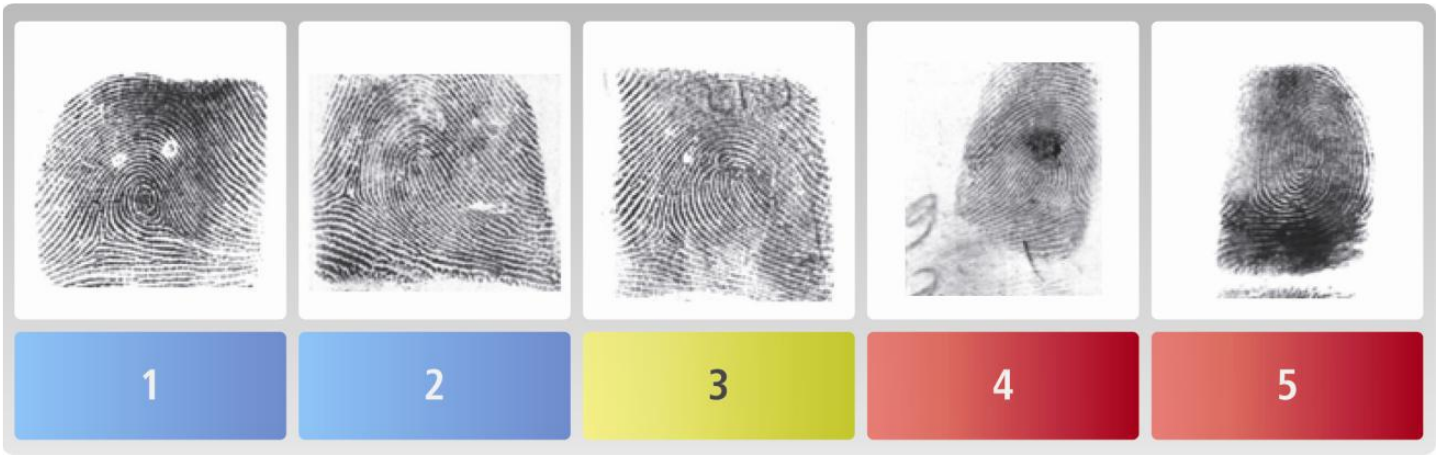


Figura 6 Clasificación de las impresiones dactilares según su calidad del Biomesys AFIS.

Los sistemas anteriormente expuestos realizan la medición de la calidad de las huellas dactilares, sin embargo no exponen en detalle las estrategias utilizadas para acometer dicho proceso. Se cuenta con las especificaciones del algoritmo NFIQ, implementado en el NBIS, y del método basado en la combinación de características locales del componente desarrollado en la UCI. Debido a que la información asociada a estas soluciones no es suficiente para conformar una propuesta que dé solución a la problemática planteada, por ello

se decide realizar un estudio de los algoritmos existentes para el cálculo de la calidad de la imagen de una huella dactilar.

1.3.2 Algoritmos para la medición de la calidad de la huella dactilar

Los métodos de medición de calidad de imágenes de huellas dactilares se clasifican en (10):

- Algoritmos basados en las características locales de la imagen
- Algoritmos basados en características globales de la imagen
- Algoritmos basados en clasificadores.

- **Métodos basados en características locales**

Los métodos basados en características locales dividen la imagen en bloques no solapados para extraer características de cada bloque. Los bloques se clasifican en grupos de diferente calidad y en función de estos se genera una medida de la calidad local. Esta última puede obtenerse a partir del porcentaje de bloques clasificados con "alta" o "baja" calidad, o una combinación elaborada. Estos algoritmos se subdividen en cinco grupos según la característica local en la que se basan (10):

1. **Basados en la dirección local:** utilizan la información de la dirección local proporcionada por el campo direccional.
2. **Basados en filtros Gabor:** los filtros Gabor pueden ser interpretados como un banco de filtros representativos de las frecuencias locales. Esta familia de filtros constituye otra implementación de los campos de dirección local.
3. **Basados en la intensidad del píxel:** utilizan la información de intensidad de los píxeles para clasificar los bloques. Hacen uso de características como la claridad y la consistencia del nivel de gris de crestas y valles.
4. **Basados en potencia del espectro:** utilizan las características del espectro, calculando la transformada discreta de Fourier de la onda sinusoidal a lo largo de la dirección de cresta local. Los bloques de baja calidad no exhiben una frecuencia dominante obvia, o está fuera del rango normal de frecuencia de crestas.
5. **Basados en la combinación de características locales:** utilizan la amplitud, frecuencia, varianza, entre otras características de las crestas y los valles para clasificar los bloques de la imagen de huella dactilar.

- **Algoritmos basados en características globales**

Los algoritmos basados en las características globales analizan la imagen de una forma integral y calculan una medida de la calidad global a partir de las características extraídas. Se clasifican en dos grupos:

1. **Basados en el campo direccional:** utilizan dos medidas básicas para analizar la estructura global de una imagen de una huella dactilar. La primera se basa en comprobar la continuidad del campo

direccional y la segunda comprueba la uniformidad del campo de frecuencia. Ambas utilizan la información de la dirección local proporcionada por el campo direccional.

2. **Basados en potencia del espectro:** utilizan los datos proporcionados por el espectro de la imagen de la huella dactilar, el cual es determinado a partir de la transformada de Fourier. Se basan en la concentración de energía en la región en forma de anillo formada en el espectro. Imágenes de alta calidad tendrán la energía concentrada en pocas bandas, mientras que las más pobres tendrán una distribución más difusa.

- **Algoritmos basados en clasificadores**

Los métodos que utilizan clasificadores definen la medida de la calidad como un grado de separación entre la distribución compatible y la no-compatible de una huella dactilar. Esto puede interpretarse como una predicción de rendimiento del módulo de comparación (10).

1.3.3 Algoritmos basados en características locales y globales

Los algoritmos basados en características locales y globales de la imagen son los más utilizados debido a su alto grado de certeza. En las tablas 1 y 2 se muestra un sumario de estos tipos de algoritmos resumidos como parte de un estudio realizado por la IEEE[9] en el año 2007.

Dirección Local
Nivel de certeza de la orientación (***OCL*** por sus siglas en inglés) Medida de fortaleza de orientación calculada del gradiente de la imagen nivelada en gris.
Frecuencia de crestas, grosor de crestas, grosor de cresta-a-valle Calculada a partir de la sinusoide que modela crestas y valles en la dirección normal a flujo de crestas.
Orientación Local Diferencia absoluta de promedios de orientación local con los bloques circundantes.
Coherencia Espacial Medida de coherencia de la dirección calculada a partir del gradiente de la imagen nivelada en gris.
Características de la simetría Correlación entre la simetría linear y parabólica en una imagen de huella dactilar.
Filtros de Gabor
Características de Gabor Desviación estándar de m filtros responden a diferentes direcciones.
Intensidad de Píxel

[9] **IEEE:** Instituto de Ingenieros Eléctricos y Electrónicos, una institución americana responsable de la creación de una gran cantidad de estándares en electrónica e informática.

Direccionalidad
Suma mínima de las diferencias de intensidad entre un píxel *(i,j)* y l píxeles seleccionados a lo largo de una segmento de línea centrado en *(i,j)*, calculada para n diferentes direcciones del segmento de línea.
Factor de Aglomeración
Grado para el cual píxeles similares se aglomeran en la región cercana.
Claridad Local
Área solapada de las distribuciones niveladas en gris de crestas y valles segmentados.
Potencia del espectro
DFT de la sinusoide que modela crestas y valles
Determina la transformada de Fourier de la onda sinusoidal que modela las crestas y valles..
Combinación de características locales
Amplitud, frecuencia y varianza de la sinusoide que modela crestas y valles
Realiza la combinación de las características locales para la medición de la calidad.
Mapa de dirección, mapa de bajo contraste, mapa de bajo flujo y mapa de curva alta
Realiza la combinación de las características locales para la medición de la calidad.

Tabla 1 Sumario de los algoritmos existentes para la medición de calidad de imágenes de huellas dactilares basados en características locales.

Campo Direccional
Continuidad del campo direccional
Detección de cambios abruptos entre los bloques.
Uniformidad del campo de frecuencia
Desviación estándar del radio de espesor entre crestas y valles.
Potencia del espectro
Concentración de energía en regiones en forma de anillos del espectro
Utilizan la información proveniente del espectro de la imagen.

Tabla 2 Sumario de algoritmos existentes para la medición de calidad de imágenes de huellas dactilares basados en características globales.

1.3.4 Principales algoritmos de medición de calidad de imágenes de huellas dactilares

- **Combinación de características locales**

El algoritmo realiza el análisis de la imagen en bloques de $16x16$ píxeles. De cada bloque se obtiene el histograma (frecuencia de aparición de cada nivel de gris de la imagen) y se determinan las características locales implicadas en el proceso. Se obtiene la media a partir de la siguiente expresión:

$$Media = \frac{1}{NM}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} I(i,j) \qquad [1]$$

La media denota el promedio de los niveles de gris de cada bloque, donde $I(i,j)$ es la intensidad de color del píxel en la fila i y la columna j del bloque I, definido por una matriz de $N\ x\ M$ dimensiones.

La varianza es una medida de la uniformidad de la imagen y se denota como:

$$Varianza = \frac{1}{NM}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}(I(i,j) - Media)^2 \quad [2]$$

La desviación estándar está definida por:

$$\sigma = \sqrt{\sum_{i=0}^{L-1}(I_i - m)^2 h(I_i)} \quad [3]$$

donde L es la matriz asociada al bloque, I_i es la intensidad del píxel, m es la media y $h(I_i)$ es el valor del histograma para esa intensidad de píxel.

Una vez calculada la desviación estándar, se obtiene la suavidad del bloque, la cual está definida por:

$$R = 1 - \frac{1}{1+\sigma^2} \quad [4]$$

La uniformidad del bloque se define por:

$$U = \sum_{i=0}^{L-1} h(I_i)^2 \quad [5]$$

Con estos datos estadísticos, se calcula lo que se conoce en inglés como *Inhomogeneity* (inH), que denota cuan no homogéneo (caótico) es el bloque que se está procesando.

$$inH = \frac{m \cdot U}{\sigma \cdot R} \quad [6]$$

Luego se calcula el contraste direccional de cada bloque determinado a partir de la aplicación de una ventana de $5x5$ píxeles. Al bloque se le asigna una de las ochos direcciones que representa la orientación de la cresta. La dirección (θ) es la mínima en el cambio de los niveles de grises.

$$\theta = min_q\left\{\sum_{x=1}^{8}\sum_{y=1}^{8} C_{xy}(q)\right\}\ para\ q = \{0,1,\dots,7\} \quad [7]$$

El término $C_{xy}(q)$ es la diferencia del nivel de gris desde el máximo al mínimo en la dirección q del filtro de $5x5$ píxeles (**Ver Anexo_4**). Denotando $\bar{\theta}$ como la dirección perpendicular a θ, el contraste mínimo y el máximo se definen como:

$$\theta_i = \sum_{x=1}^{8}\sum_{y=1}^{8} C_{xy}(\theta) \quad [8]$$

$$\bar{\theta}_i = \sum_{x=1}^{8}\sum_{y=1}^{8} C_{xy}(\bar{\theta}) \quad [9]$$

donde i es el índice del bloque. Luego, la calidad es definida como:

[10]

$$Q = \frac{\sum_{i=1}^{N}(\theta_i - \bar{\theta}_i)}{N \sum_{x=1}^{8}\sum_{y=1}^{8} C} \cdot 100$$

donde N es el número de bloques y C una constante.

Esta medida se le aplica a los bloques clasificados de "buena calidad" en el procesamiento de los pasos anteriores, definiéndose umbrales en cada dato estadístico que se obtiene.

La inH es aceptable por debajo de 2000, la varianza debe ser superior a 1000 y la desviación estándar debe estar por encima de 100. A los bloques que estén dentro de estos umbrales, se le aplica el contraste direccional, y si el resultado está por encima de 70, se toma como de buena calidad. Dependiendo de la cantidad de bloques que sean buenos, se da una clasificación entre 1 y 5, definiendo la calidad 1 como la mejor y la 5 como la peor.

El algoritmo es definido en (14) como parte de las especificaciones del componente desarrollado en la UCI.

- **Análisis local: combinación de OCL y estructura de cresta-valle**

Este método, definido en (17), representa una combinación del algoritmo *OCL* y el basado en la estructura cresta-valle. Para el análisis de la estructura local, las imágenes son divididas en bloques de $32x32$ píxeles

Para determinar el nivel de certeza de la orientación se usa el gradiente de la imagen en niveles de gris. El gradiente de nivel de gris (dx, dy) en un píxel representa la orientación y la fuerza de la orientación de la imagen en este píxel.

Mediante la aplicación del algoritmo PCA[10] sobre los gradientes de la imagen en un bloque, una base ortogonal puede formarse mediante la búsqueda de sus valores y vectores propios. La relación entre los dos valores propios da una indicación de cuan fuerte que se concentra la energía a lo largo de la dirección dominante con dos vectores apuntando a la dirección normal y tangencial del flujo promedio de cresta respectivamente. La matriz de covarianza C del vector gradiente de un bloque de la imagen de N puntos está dado por:

$$C = E\left\{\begin{bmatrix} dx \\ dy \end{bmatrix} [dx \;\; dy]\right\} = \begin{matrix} a & c \\ c & b \end{matrix} \qquad [11]$$

donde:

$$E\{*\} = \frac{1}{N}\sum_{N} * \qquad [12]$$

Para la matriz de covarianza C, los valores propios λ se definen como:

$$\lambda_{max} = \frac{(a+b) + \sqrt{(a-b)^2 + 4c^2}}{2} \qquad [13]$$

$$\lambda_{min} = \frac{(a+b) - \sqrt{(a-b)^2 + 4c^2}}{2} \qquad [14]$$

Para un bloque de imagen de la huella, la razón entre λ_{min} y λ_{max} es:

---

[10] **PCA:** el análisis de componentes principales (*Principal Component Analysis* en inglés) es una técnica utilizada para reducir las dimensiones de un conjunto de datos.

$$ocl = \frac{\lambda_{min}}{\lambda_{max}} = \frac{(a+b) - \sqrt{(a-b)^2 + 4c^2}}{(a+b) + \sqrt{(a-b)^2 + 4c^2}} \quad [15]$$

El nivel de certeza de la orientación representa la concentración de la energía a lo largo de la orientación cresta-valle de un bloque. El valor ocl está entre 0 y 1, y tanto a como b son mayores que 0.

Se calcula, a partir del gráfico de niveles de gris para un bloque de imagen en la dirección normal al flujo de cresta, la frecuencia y el grosor de las crestas, así como la razón de grosor de cresta-valle. Dado que en una huella dactilar humana no pueden aparecer crestas extremadamente cerca o separadas, se usa el valor de frecuencia nominal de cresta como garantía de que la imagen capturada es buena. Del mismo modo, las crestas que son irrazonablemente gruesas o delgadas indican que la imagen de la huella no se capturó correctamente o es una imagen residual.

Se determinan umbrales para valores válidos del nivel de certeza de la orientación, la frecuencia y el grosor de las crestas. Dichos umbrales permiten clasificar la certeza de la orientación y la estructura cresta-valle del bloque de la imagen en “buena”, “mala” o “indeterminada”. Además se define un umbral adaptativo que identifique el bloque como “en blanco”. La tabla 3 resume las clasificaciones de calidad para un bloque según su certeza de orientación y su estructura cresta-valle.

Estructura cresta-valle / Certeza de La orientación	Buena	Indeterminada	Mala
Buena	Bueno	Bueno	Indeterminado
Indeterminada	Bueno	Indeterminado	Malo
Mala	Indeterminado	Malo	Malo

Tabla 3 Clasificación de bloques según la certeza de orientación y estructura cresta-valle.

La medida de calidad total resultante del análisis local está dada por:

$$S_L = \frac{T_G + 0.5 \cdot T_U}{T_G + T_U + T_B} \quad [16]$$

donde T_G, T_Uy la T_B se refieren al número total de bloques de imágenes de calidad buena, indeterminada y mala respectivamente. El valor S_L está entre 0 y 1, donde 0 es representa la calidad más baja y 1 la mayor calidad.

- **Método basado en características locales y globales**

Esta estrategia es definida por Chaohong, Tulyakov y Govindaraju en (18), y determina la calidad de una imagen de huella dactilar a través de la combinación de características locales y globales. Utiliza un algoritmo basado en la concentración de la energía del espectro en la región en forma de anillo para estimar la calidad a nivel global de la imagen, mientras que para estimar la calidad teniendo en cuenta características locales, hace

uso de un algoritmo basado en la no homogeneidad con contraste direccional. Según los valores obtenidos, tanto a nivel global como local, se clasifica la calidad de la imagen siguiendo determinados parámetros. Este método establece cinco niveles de calidad:

- **Nivel 1** - (buena) contraste claro, crestas detectadas fácilmente, minucias localizadas con precisión, segmentado fácil.
- **Nivel 2** - (normal) la mayor parte de las crestas pueden ser detectadas; contraste medio, buena cantidad de minucias, posee algunos bloques de mala calidad (secos o manchados).
- **Nivel 3** - (manchada / húmeda) crestas mal separadas.
- **Nivel 4** - (seca / ligeramente entintadas) crestas rotas; sólo una pequeña parte de las crestas se pueden separar.
- **Nivel 5** - (estropeada) crestas totalmente corruptas.

Para determinar la medida de calidad global de la imagen se usa un algoritmo basado en la energía espectral limitada en una región en forma de anillo**.** Debido al patrón de direccionalidad de una imagen de huella dactilar, esta es representada a través del espectro de Fourier.

Las características del espectro FFT[11] se pueden simplificar al expresarlas en coordenadas polares, por lo que se representa el espectro con la función $S(r,\theta)$, donde r es la distancia radial desde el origen y θ es la variable angular. Si se define que $fft2$ representa la función de la transformada discreta de Fourier en 2-D (dos dimensiones) y $fftshift$ mueve el origen de la transformada al centro del rectángulo de frecuencia, entonces el espectro de FFT $S(r,\theta)$ se expresa como:

$$S(r,\theta) = \log\left(1 + abs\left(fftshift\left(fft2\left(img\right)\right)\right)\right) \qquad [17]$$

Se convierte $S(r,\theta)$ a la función de 1-D (una dimensión) $S_\theta(r)$ para cada dirección, y se analiza para un ángulo determinado. Como resultado, se obtiene el perfil del espectro a lo largo de una dirección radial desde el origen. Luego, un descriptor global se determina mediante la suma de las variables discretas:

$$S(r) = \sum_{\theta=0}^{\pi} S_\theta(r) \qquad [18]$$

Sobre la base de los cálculos y análisis actuales de patrones, se calcula la energía de banda entre las frecuencias 30 y 60, denominada “medida espectral de la región limitada en forma de anillo”. La diferencia entre imágenes de buena y mala calidad es significativa, dada por la existencia de un pico principal característico bien definido y una mayor distribución de energía en las imágenes de buena calidad. La figura 7 muestra esta diferencia a través de dos impresiones de un mismo dedo, una de buena calidad y una de mala calidad. En la figura, (a) y (b) corresponden al espectro y el espectro de la región en forma de anillo, respectivamente, de la impresión de buena calidad, (c) y (d) están asociados a la imagen de mala calidad.

[11] **FFT:** siglas correspondientes a la transformada rápida de Fourier (*Fast Fourier Transform*). La transformada de Fourier convierte el tiempo (o espacio) en frecuencia y viceversa.

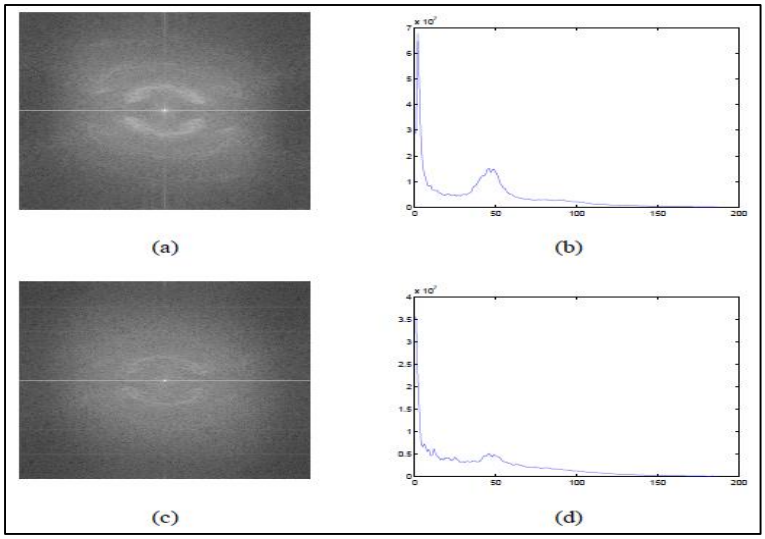


Figura 7 Medidas espectrales de impresiones dactilares.

Para cuantificar la condición, desde el punto de vista local, de las imágenes de huellas dactilares, el método hace uso de las propiedades estadísticas del histograma de intensidad. Se define que I_i, L, y $H(I)$ representan la intensidad de nivel de gris, el número de posibles intensidades de nivel de gris y el histograma de los niveles de intensidad respectivamente. La media (m) y la desviación estándar (σ) se denotan como se muestra:

$$m = \sum_{i=0}^{L-1} I_i h(I_i) \qquad [19]$$

$$\sigma = \sqrt{\sum_{i=0}^{L-1} (I_i - m)^2 h(I_i)} \qquad [20]$$

La suavidad (R), la uniformidad (U) y el índice de no homogeneidad de un bloque (inH) se determinan a partir de las ecuaciones [4], [5] y [6] respectivamente.

El contraste direccional representa la certeza de la orientación de flujo de cresta local e identifica las regiones en peor estado. Para determinar el contraste local se determina, para cada píxel, la suma de los valores del píxel en ocho direcciones en una vecindad de $9x9$ (S_i). Los valores S_{max} y S_{min} corresponden a las direcciones más probables de píxeles blancos en los valles y píxeles negros en las crestas respectivamente. Al calcular el promedio de los valores correspondientes a la razón S_{min}/S_{max} de los píxeles del bloque, se obtiene una medida del contraste direccional. A través de un examen visual se determina un umbral para este promedio, si este último es mayor que el umbral, entonces el bloque no tiene buen contraste direccional. Las minucias detectadas en estas áreas o cercanas a ellas son eliminadas como falsas minucias.

Este método define que un bloque es **bueno** si su inH es menor que 10 y el promedio de contraste es mayor que 50. Un bloque es **húmedo** si el producto de su media y su desviación estándar es menor que un umbral definido. Un bloque es **seco** si su media es mayor que un umbral determinado, su promedio de contraste está

entre 20 y 50, la razón entre su media y su promedio de contraste es mayor que 5, y la razón entre su uniformidad y su suavidad (U/R) es mayor que 20.

El método establece que:

- Si el porcentaje de bloques con un contraste direccional muy bajo está por encima del 30%, la imagen se clasifica como nivel 5.
- Si la energía espectral limitada en forma de anillo está por debajo del umbral S_l y el porcentaje de bloques buenos está por debajo del 30%, la imagen se clasifica de nivel 4 si el porcentaje de bloques secos está por encima de 30% y es de nivel 3 si el porcentaje de bloques húmedos está por encima de 30%.
- Las imágenes de nivel 1 poseen alta energía espectral en la región limitada en forma de anillo y más del 75% de bloques buenos. Las imágenes de nivel 2 tienen una energía espectral media y menos del 75% de bloques buenos.

- **Algoritmo basado en la coherencia local del campo de orientación**

El algoritmo, definido en (3), determina la coherencia local del campo de orientación a través del cálculo de gradientes en la imagen de la huella dactilar. Se define que I es una imagen de huella dactilar en escala de grises con g niveles de gris y $I[x,y]$ es el nivel de gris del píxel $[x,y]$ en I. La orientación de cresta local en un píxel $[x,y]$ está dada por el ángulo θ_{xy} que las crestas de una huella dactilar, cruzando a través de una pequeña vecindad arbitraria centrada en $[x,y]$, forman con el eje horizontal.

Una imagen de orientación de una huella dactilar (también llamada imagen direccional), es una matriz **D** cuyos elementos representan la orientación local de las crestas de la huella. Cada elemento θ_{ij}, correspondiente al nodo $[i,j]$ de una red de malla cuadrada situada sobre el píxel $[x_i, y_j]$, indica la orientación promedio de las crestas en una vecindad de $[x_i, y_i]$. El valor r_{ij} se asocia a cada elemento θ_{ij} para denotar la fiabilidad (o consistencia) de la orientación.

El gradiente $\nabla(x,y)$ en el punto $[x,y]$ de I, es un vector bidimensional $[\nabla_x(x,y),\nabla_y(x,y)]$ donde los componentes ∇_x y ∇_y representan las derivadas de I en $[x,y]$ con respecto a las direcciones x y y respectivamente. Siguiendo el método propuesto por Ratha, Chen y Jain en (19), se calcula la orientación dominante de la cresta a través de la combinación de múltiples estimaciones de gradientes en una vecindad W de $17x17$ centrada en $[x_i, y_j]$:

$$\theta_{ij} = 90^{\circ} + \frac{1}{2}\operatorname{atan2}\left(2G_{xy,}G_{xx} - G_{xy}\right) \quad [21]$$

$$G_{xy} = \sum_{h=-8}^{8}\sum_{k=8}^{8} \nabla_x(x_i + h, y_i + k) \cdot \nabla_x(x_i + h, y_i + k)$$

$$G_{xx} = \sum_{h=-8}^{8}\sum_{k=8}^{8} \nabla_x(x_i + h, y_i + k)^2$$

$$G_{yy} = \sum_{h=-8}^{8} \sum_{k=8}^{8} \nabla_y (x_i + h, y_i + k)^2$$

En las fórmulas anteriores ∇_x y ∇_y son los componentes de $x - gradiente$ y $y - gradiente$ calculados a través de máscaras de Sobel[12] de $3x3$, y $atan2(y, x)$ calcula el arco-tangente de las variables y y x. El cálculo de $atan2(y, x)$ es similar al proceso de calcular el arco-tangente de y/x, excepto que los signos de ambos argumentos son usados para determinar el cuadrante del resultado.

La consistencia o fiabilidad r de la estimación θ se deriva de la concordancia o coherencia. Siguiendo el enfoque basado en el gradiente, se determina la coherencia según la siguiente ecuación:

$$r_{ij} = coherencia(\theta_{ij}) = \frac{\sqrt{(G_{xx} - G_{yy})^2 + 4{G_{xy}}^2}}{G_{xx} + G_{yy}} \qquad [22]$$

El valor r_{ij} es bajo para las regiones de baja calidad y alto para las regiones de buena calidad de la imagen de la huella.

- **Análisis global: algoritmo basado en el espectro de Fourier**

El algoritmo utiliza el espectro de Fourier para estimar la calidad global de una imagen de huella dactilar. De acuerdo con su descripción en (20), se busca en primer lugar la banda de frecuencia correspondiente al período promedio global de la cresta. Luego el nivel de calidad de la imagen de la huella se calcula midiendo la magnitud relativa de los componentes de frecuencia de banda.

Para las funciones discretas (imágenes) de dos dimensiones, las transformadas discretas directa e inversa de Fourier se definen respectivamente como:

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) exp\left[-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)\right] \qquad [23]$$

$$F(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} f(u, v) exp\left[j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)\right] \qquad [24]$$

donde $f(x, y)$ es una imagen de MxN. El espectro de Fourier de las imágenes se define como:

$$|F(u, v)| = \sqrt{R(u, v)^2 + I(u, v)^2} \qquad [25]$$

donde $R(u, v)$ y $I(u, v)$ son las partes real e imaginaria de la transformada de Fourier respectivamente. Cada píxel en el espectro de Fourier representa un cambio de intensidad en la frecuencia espacial de un ciclo por ancho de la imagen.

Para el cálculo de la magnitud relativa de los componentes de frecuencia de banda se ubica en primer lugar el anillo del espectro. Para una imagen de huella dactilar de tamaño MxM, la geometría del anillo en el espectro

[12] **Máscaras de Sobel:** este término está asociado a las matrices resultantes de la aplicación del filtro de Sobel, el cual es un filtro de aproximación al gradiente que detecta los bordes horizontales y verticales separadamente sobre una imagen en escala de grises.

[26]

es similar a un círculo, por lo que únicamente se requiere calcular el radio del anillo a través del siguiente operador:

$$max(R)\left\{G_\sigma(R) * \oint \frac{p(u,v)}{2\pi R} ds\right\}$$

donde $p(u,v)$ es una imagen de potencia de espectro de una huella dactilar, ***** denota la convolución y $G_\sigma(R)$ es una función de suavizado, como una gaussiana de escala σ. El operador completo se comporta como un detector circular difuminado a una escala fijada por σ. Para las imágenes de MxM, los intervalos de cresta-valle (período) está entre 5 y 23 píxeles, y el rango de los radios del anillo, en el espectro, está entre $[M/23, M/5]$.

En las imágenes de tamaño MxN, en lugar de un anillo circular, se muestra un anillo elíptico de magnitud mayor en el espectro. Este anillo es localizado siguiendo el siguiente operador:

$$max(R)\left\{G_\sigma(R) * \int_0^\pi \oint p(u,v)\, d\theta\right\} \quad [27]$$

Se define que:

$$u = \frac{M}{N} r \sin\theta \quad [28]$$

$$v = r\cos\theta \quad [29]$$

donde r es la mitad de la distancia entre los puntos máximos de la elipse sobre el eje vertical. Dado que el radio del anillo está en proporción directa a la frecuencia de cresta y en proporción inversa al período de cresta, el promedio global del período de cresta puede estimarse como $P = N / r$. Se calcula la calidad $\boldsymbol{QS}$ de la imagen como sigue:

$$QS = C\frac{\int_{\frac{5}{6}r \le R \le \frac{6}{5}R} \oint \frac{P(u,v)}{2\pi r} dsdr}{\int_{r \ge \frac{1}{4}R} \oint \frac{P(u,v)}{2\pi r} dsdr} \quad [30]$$

donde R es el radio del anillo, C es un factor de normalización encargado de normalizar a QS en un rango de 0 a 1. El algoritmo clasifica la calidad de la imagen en “buena”, “normal” o “pobre”, donde 1 es el valor máximo (calidad buena) y 0 es el mínimo (calidad pobre).

- **Algoritmo basado en las características de la simetría**

Según su descripción en (21), el algoritmo parte de que el tensor de orientación de una imagen permite obtener indicadores de calidad como el ruido, la falta de estructura y el desenfoque, por lo que se apoyan en este para determinar un set de características asociadas a la simetría, ofreciendo finalmente una vía de determinación de la calidad de la imagen de una huella dactilar.

El objetivo del algoritmo es determinar si la información del tensor de orientación se estructura en cierto sentido, esto se hace con el fin de distinguir el contenido ruidoso de estructuras posiblemente no-triviales. El algoritmo se basa en descomponer el tensor de orientación de una imagen en representaciones de simetría,

[31]

donde las simetrías incluidas están relacionadas con la definición particular de calidad. El tensor está dado por la expresión:

$$z = (D_x f + iD_y f)^2$$

donde $D_x f$ y $D_y f$ denotan las derivadas parciales de la imagen en los ejes x y y. Para el cálculo de las derivadas se utilizan gaussianas separables con una pequeña desviación estándar. Luego el tensor de orientación se descompone en elementos de la simetría de orden n, donde la n-ésima simetría está dada por $\exp(in\emptyset + \alpha)$. Los filtros que modelan estas descripciones de simetrías se obtienen a partir de:

$$h_n = (x + iy)^n \cdot g \text{ para } n \geq 0$$
$$h_n = (x - iy)^{|n|} \cdot g \text{ para } n < 0 \qquad [32]$$

donde g denota una gaussiana de dos dimensiones con desviación estándar (σ).

Para descomponer una imagen en determinadas simetrías se calcula $\langle z, h_n \rangle$, donde $\langle , \rangle$ representa el producto escalar en dos dimensiones, produciendo respuestas complejas asociadas a $s_n = c \cdot \exp(i\alpha)$, donde c es la certeza de ocurrencia y α codifica el patrón de orientación de simetría n (para $n \neq 2$). Las respuestas de filtro normalizadas se obtienen calculando $\hat{s}_n = \frac{\langle z, h_n \rangle}{\langle |z|, h_0 \rangle}$, es decir, dividiendo s_n entre la cantidad de certeza. De esta manera $\{\hat{s}_n\}$ describe las propiedades de simetría de una imagen en términos de n órdenes.

Se calcula la covarianza de $\{|\hat{s}^i{}_n|\}$ en bloques de tamaño bxb. Un valor negativo elevado de la covarianza es codiciado en términos de calidad, ya que esto sugiere simetrías bien separadas con medidas confiables de certidumbre c. Por otro lado, la covarianza positiva implica la co-ocurrencia de tipos de simetría mutuamente excluyentes en las proximidades de un punto, lo que es considerado una indicación de ruido o desenfoque. Esta información se incorpora al algoritmo mediante la ponderación de la certeza de la simetría.

Sumando $\{\hat{s}^i{}_n\}$ en cada píxel proporciona un total de simetría s_{total}, el cual es promediado en bloques de bxb dando lugar a s_b. La medida de calidad q_b de cada bloque está dada por:

$$q_b = y(|r_b|) \cdot \chi(-r_b) \cdot s_b \qquad [33]$$

donde χ representa la función Heavyside (1 para argumentos positivos, 0 en caso contrario) y r_b denota el coeficiente de correlación entre $\{|\hat{s}^i{}_n|\}$ para el bloque b. El elemento r_b se calcula promediando los coeficientes de correlación entre dos órdenes $r_{b_{k,l}}$ involucrados:

$$r_{b_{k,l}} = \frac{Cov(|s^i{}_k|, |s^i{}_l|)}{Var(|s^i{}_k|)Var(|s^i{}_l|)} \qquad [34]$$

Una función de mapeo y, escogida empíricamente, controla la influencia de r_b. Se propone el uso de la función $y(t) = t^2$ ya que hace que el método sea muy estricto. Una métrica final Q se establece a través del promedio de q_b de los bloques "de interés" i_b, los cuales están representados por los bloques donde $s_b > \tau$, teniendo así una respuesta de simetría total minina. Esta métrica cae en el intervalo $[0, Q_{max}]$ para alguna $Q_{max} \leq 1$, dependiendo de la función de mapeo y escogida.

1.4 Metodologías de desarrollo de software

Una metodología de desarrollo, en ingeniería de software, es un conjunto de herramientas, técnicas, procedimientos y soporte documental encaminados a estructurar, planificar y controlar el proceso de desarrollo de forma organizada y lógica, que tienen como objetivo apoyar a los desarrolladores en la creación de un nuevo software (22).

Las metodologías de desarrollo se clasifican en dos clases: las metodologías tradicionales o robustas y las ágiles o ligeras. En la Tabla 5 se presenta una comparación entre los dos tipos de metodologías (23).

- Metodologías tradicionales

Las metodologías tradicionales o prescriptivas definen un conjunto de actividades, acciones, tareas, fundamentos y productos de trabajo que se requieren para desarrollar software de alta calidad. El proceso conduce a un equipo de software a través de un conjunto de actividades del marco de trabajo que se organizan en un flujo de proceso, el cual puede ser lineal[13], incremental[14] o evolutivo[15]. La terminología y los detalles de cada modelo difieren, pero las actividades genéricas del marco de trabajo permanecen razonablemente consistentes (24).

- Metodologías ágiles

Las metodologías ágiles combinan una filosofía y conjunto de directrices de desarrollo. La filosofía busca la satisfacción del cliente y la entrega temprana de software incremental, equipos de proyecto pequeños y con alta motivación, métodos informales, un mínimo de productos de trabajo de la ingeniería de software, y una simplicidad general de desarrollo (24). Las directrices de desarrollo resaltan la entrega sobre análisis y diseño (aunque estas actividades no se descartan) y la comunicación activa entre los desarrolladores y los clientes.

En febrero del 2001 queda conformado durante una reunión en Utah, Estados Unidos, el Manifiesto Ágil. Este manifiesto recogía los principales principios y particularidades de este tipo de metodología, y afirmaba que en esta se valora (25):

- Al individuo y las interacciones del equipo de desarrollo, sobre el proceso y las herramientas.
- Desarrollar software que funcione, más que conseguir una buena documentación.
- La colaboración con el cliente, más que la negociación de un contrato.
- Responder a los cambios, más que seguir estrictamente un plan (23).

---

[13] **Flujo lineal:** está en función de aquellos modelos que desarrollan sus actividades de forma continua sin retrocesos a actividades previas y sin repetición de las ya ejecutadas.

[14] **Flujo incremental:** asociado a modelos que responden a una situación donde los requisitos están bien definidos pero es necesario satisfacer al cliente de forma rápida con conjuntos limitados de funcionalidad en pequeñas porciones que aumentan gradualmente y se refinan y expanden en cada entrega.

[15] **Flujo evolutivo**: vinculado a modelos que responden a una situación donde los requisitos finales no están bien definidos, aunque sí el esquema general de necesidades del cliente pero es necesario satisfacer al cliente de forma rápida ante la presión del mercado, dedicándose inicialmente a satisfacer requisitos esenciales y luego trabajar sobre las extensiones de estos requisitos.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de software.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparado para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo de desarrollo).	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas y normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (menos de 10 estudiantes) trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos
Pocos artefactos.	Más artefactos.
Pocos riles.	Más roles.
Menos énfasis en la arquitectura de software.	La arquitectura de software es esencial y se expresa mediante modelos.

Tabla 4 Comparación entre metodologías tradicionales y ágiles.

- **RUP**

El **Proceso Unificado de Desarrollo** (RUP) es un proceso de software dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. El proceso utiliza el Lenguaje Unificado de Modelado (UML). Pone en práctica el basar gran parte del proyecto de desarrollo en componentes reutilizables, es decir, en piezas de software con una interfaz bien definida. Propone un levantamiento exhaustivo de requerimientos, e intenta reducir el número de cambios realizando un análisis y diseño tan completo como sea posible. Las necesidades de los clientes no son fáciles de discernir, por lo que existe un contrato prefijado con los mismos, los cuales interactúan con el equipo de desarrollo mediante reuniones. Promueve la documentación abundante, explícita y detallada (26). La metodología RUP divide el desarrollo de software en cuatro fases:

• **Inicio:** el objetivo en esta etapa es determinar la visión del proyecto.

• **Elaboración:** en esta etapa el objetivo es determinar la arquitectura óptima.

• **Construcción:** en esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.

• **Transición:** el objetivo es finalizar el proyecto y transferir el software al usuario final (27).

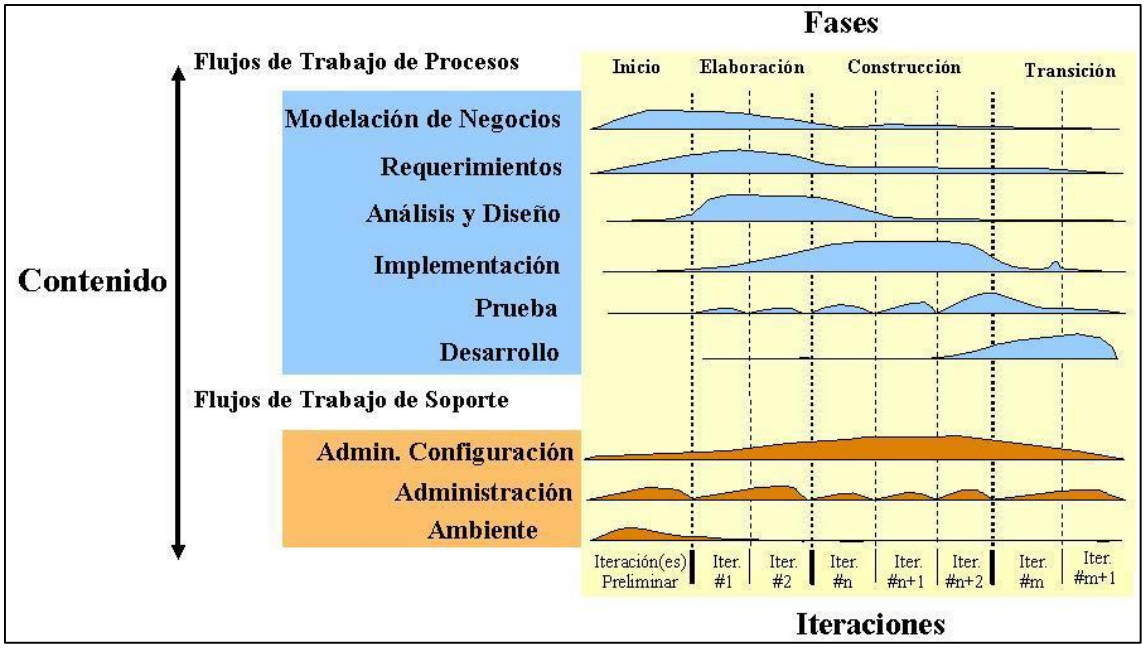


Figura 8 Fases y flujos de trabajo establecido por RUP.

- **Extreme Programming (XP)**

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (23).

El ciclo de vida propuesto por la metodología se divide en las siguientes fases:

1) **Fase de exploración:** los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.
2) **Fase de planificación:** el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, además de las entregas relacionadas con éstas. El resultado de esta fase es un Plan Entregas.
3) **Fase de iteraciones:** esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entregas está compuesto por las diferentes iteraciones. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto,

escogiendo las historias de usuario que fuercen la creación de esta arquitectura, sin embargo, no siempre es posible ya que es el cliente quien decide que historias se implementarán en cada iteración. Al final de la última iteración el producto estará listo para entrar en producción.

4) **Fase de puesta en producción:** en esta fase se entregan módulos funcionales y sin errores, y según los intereses del cliente el sistema puede ponerse en producción o no. No se realizan desarrollos funcionales, pero pueden ser necesarias tareas de ajuste.
5) **Fase de mantenimiento:** mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para esto se requiere de tareas de soporte para el cliente. La velocidad de desarrollo puede disminuir después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.
6) **Fase de muerte del proyecto:** el cliente no tiene más historias para ser incluidas en el sistema, haciendo necesario que se satisfagan sus necesidades en otros aspectos tales como rendimiento y confiabilidad del mismo. Se genera la documentación final y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

La principal idea asociada a XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto. XP apuesta por un crecimiento lento del costo del cambio y con un comportamiento asintótico. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las siguientes prácticas:

1) **El juego de la planificación:** el alcance de la siguiente versión está definido por consideraciones del negocio (prioridad de los módulos, fechas de entrega) y estimaciones técnicas (estimaciones de funciones, consecuencias). El objetivo del juego es maximizar el valor del software producido.
2) **Versiones pequeñas:** un sistema simple se pone rápidamente en producción. Periódicamente se producen nuevas versiones agregando en cada iteración aquellas funciones consideradas valiosas para el cliente.
3) **Metáfora del sistema:** cada proyecto es guiado por una historia simple que explica el funcionamiento del sistema en general, reemplaza a la arquitectura y debe estar en lenguaje común, entendible para todos, cliente y desarrolladores.
4) **Diseño simple**: el sistema se diseña con la máxima simplicidad posible. Se plasma el diseño en las tarjetas CRC (Clase - Responsabilidad - Colaboración), con lo que las clases definidas durante el análisis pueden ser filtradas para determinar las que son realmente necesarias para el sistema.
5) **Pruebas continuas:** los casos de prueba se escriben antes que el código. Los desarrolladores escriben pruebas unitarias y los clientes especifican pruebas funcionales.

6) **Refactorización:** es posible reestructurar el sistema sin cambiar su comportamiento, por ejemplo eliminando código duplicado, simplificando funciones.
7) **Programación por parejas:** el código es escrito por dos personas trabajando en la misma computadora.
8) **Posesión colectiva del código:** nadie es dueño de un módulo, cualquier programador puede cambiar cualquier parte del sistema en cualquier momento, por ello siempre se deben utilizar estándares de codificación.
9) **Integración continua**: los cambios se integran en el código base varias veces por día. Todos los casos de prueba deben ejecutarse antes y después de la integración. Se dispone de una máquina para la integración y se realizan pruebas funcionales donde participa el cliente.
10) **Cliente en el sitio:** el equipo de desarrollo tiene acceso todo el tiempo al cliente, el cual está disponible para responder preguntas y fijar prioridades.
11) **Estándares de codificación:** todo el código debe estar escrito de acuerdo a un estándar de codificación.

- **Scrum**

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. El desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. Otra característica importante son las reuniones a lo largo proyecto. Estas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (28).

Scrum es una metodología ágil, y como tal (29):

- Es un modo de desarrollo de carácter adaptable más que predictivo.
- Orientado a las personas más que a los procesos.
- Emplea la estructura de desarrollo ágil: incremental basada en iteraciones y revisiones.

1.4.1 Selección de la metodología a utilizar

Luego del análisis de las características de diferentes metodologías de desarrollo de software, se decide que XP sea la que dirija el proceso de desarrollo del componente en cuestión puesto que:

- Esta metodología consiste en una programación priorizada donde el usuario final forma parte del equipo de desarrollo, uno de los requisitos para alcanzar el éxito del proyecto y la satisfacción del cliente.

- Con el uso de esta metodología se simplifica, pero no se descarta, el diseño, con el objetivo de agilizar el proceso de desarrollo.
- Todo el proceso de esta metodología está encaminado a conseguir la meta final, el software, otorgándole gran importancia a las relaciones interpersonales de los miembros del equipo de desarrollo, así como a la velocidad de reacción ante los constantes cambios que puedan surgir durante el desarrollo.

1.5 Herramientas y tecnologías

1.5.1 Lenguaje para el modelado

En el proceso de desarrollo de software se hace de carácter fundamental e imprescindible el modelado del mismo, este le proporciona al desarrollador un conjunto de herramientas, artefactos y notaciones que le garantizan una "idea visual" del sistema en construcción, lográndose una lógica de este último.

- **UML**

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, medios y dominios de aplicación. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar.
UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos (30).

1.5.2 Herramientas para el diseño

Las herramientas **CASE** (***C****omputer* ***A****ided* ***S****oftware* ***E****ngineering*, o Ingeniería de Software Asistida por Computadora) constituyen aplicaciones informáticas dirigidas a aumentar la productividad en el desarrollo de software, reduciendo el costo del mismo en términos de tiempo y de dinero. Estas herramientas pueden soportar todos los aspectos del ciclo de vida de desarrollo del software en tareas como diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores (31).

- **Rational Rose**

Rational Rose es una herramienta de modelado visual para apoyar el análisis y diseño de sistemas del software orientados a objeto. Usando el modelado, se pueden atrapar desperfectos del diseño a tiempo, mientras no sean costosos de reparar. Describe con lujo de detalles lo que el sistema incluirá y cómo trabajará, así que los desarrolladores pueden usar el modelo como una heliografía para el sistema en construcción. Incluye todos los diagramas UML (32).

- **Visual Paradigm**

Visual Paradigm es una herramienta CASE para el modelado de lenguaje UML que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite construir todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (33).
Visual Paradigm cumple con las políticas actuales de migración a software libre, siendo multiplataforma, de forma tal que facilita la modelación del software independientemente del sistema operativo que se emplee (34).

1.5.3 Lenguajes de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar procesos que pueden ser ejecutados por máquinas computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (35).

- **C++**

Hacia el año 1979 se crea por Bjarne Stroustrup una versión experimental denominada “C con clases” (“C with Classes” en inglés) con la intención de proporcionar una herramienta de desarrollo para el núcleo UNIX en ambientes distribuidos. Ya en 1983 se rebautiza como C++ y en 1985 Stroustrup publica la primera edición del libro “The C++ Programming Language” que sirvió de estándar informal y texto de referencia.
Desde sus inicios, C++ intentó ser un lenguaje que incluyera completamente al lenguaje C, pero al mismo tiempo incorpora muchas características sofisticadas, tales como: Programación Orientada a Objetos, tratamiento de excepciones, sobrecarga de operadores y plantillas (36).
Como lenguaje C, C++ adopta una visión muy cercana al lenguaje máquina. En un principio se destinó a escribir sistemas operativos, pero sus características le abrieron perspectivas nuevas. El lenguaje está formado por instrucciones muy explicitas, cortas, cuya duración de ejecución puede preverse con antelación, en el momento de escribir el programa (37).
Los programas hechos en C++ son rápidos, portables, flexibles y versátiles. Tiene características de lenguaje de alto y bajo nivel (38).

- **Java**

El lenguaje Java fue denominado originalmente "Oak". Sus inicios datan de 1991 cuando James Gosling, de la compañía Sun Microsystems, encabezó un proyecto cuyo objetivo original era implementar una máquina virtual ampliamente portable y un lenguaje de programación, ambos orientados a dispositivos "embebidos[16]". El lenguaje en sí mismo toma mucha de su sintaxis de Lenguaje de Programación C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria (36).

Sus características más notables son:

1. Simple: basado en el lenguaje C++ pero donde se eliminan muchas de las características OOP[17] que se utilizan esporádicamente.
2. Orientado al objeto.
3. Java da buen soporte a las técnicas de desarrollo OOP y en resumen a la reutilización de componentes de software.
3. Distribuido**:** Java se ha diseñado para trabajar en ambiente de redes y contienen una gran biblioteca de clases para la utilización del protocolo TCP/IP, incluyendo HTTP y FTP.
4. Interpretado: El compilador Java traduce cada fichero fuente de clases a código de bytes (*Bytecode*), que puede ser interpretado por todas las máquinas que den soporte a un visualizador de que funcione con Java.
5. Sólido: el código Java no se quiebra fácilmente ante errores de programación. En Java no es posible escribir en áreas arbitrarias de memoria ni realizar operaciones que corrompan el código.
6. Seguro: Las mismas características antes descritas que evitan la corrupción de código evitan su manipulación.
7. Multihilos: Java puede aplicarse a la realización de aplicaciones en las que ocurra más de una cosa a la vez.
8. Dinámico: Al contrario que C++ que exige se compile de nuevo la aplicación al cambiar una clase madre Java utiliza un sistema de interfaces que permite aligerar esta dependencia. Como resultado, los programas Java pueden permitir nuevos métodos y variables en un objeto de biblioteca sin afectar a los objetos dependientes (39).

- **C#**

C# es un lenguaje orientado a objetos simple, elegante y con seguridad que permite generar una gran variedad de aplicaciones. Proporciona mecanismos intrínsecos de código de confianza para obtener un alto nivel de

[16] **Dispositivos embebidos:** son procesadores incorporados en diversos dispositivos de consumo masivo como tostadoras, PDA y teléfonos móviles.

[17] **OOP:** Programación orientada a objetos (OOP por sus siglas en ingles).

seguridad, la recolección de elementos no utilizados y la seguridad de tipos. Admite herencia única y crea lenguaje intermedio de Microsoft como entrada de compiladores de código nativo (40).

Está completamente integrado con .NET Framework y Common Language Runtime[18], que conjuntamente proporcionan interoperabilidad del lenguaje, recolección de elementos no utilizados, seguridad ampliada y compatibilidad de versiones mejorada. C# simplifica y moderniza algunos de los aspectos más complejos de C y C++, como los espacios de nombres, las clases, las enumeraciones, la sobrecarga y el control estructurado de excepciones. C# también elimina ciertas características de C y C++ como macros, herencia múltiple y clases base virtuales. Para los programadores de C++ actuales, C# proporciona un lenguaje alternativo de gran potencia y productividad.

Visual C# proporciona prototipos de algunos de los tipos de proyectos más comunes, incluyendo:

- Aplicación para Windows.
- Biblioteca de clases.
- Biblioteca de control de Windows.
- Aplicación Web ASP.NET.
- Servicio Web ASP.NET.
- Biblioteca de control Web.
- Aplicación de consola.
- Servicio de Windows (41).

1.5.4 IDE

Se define como IDE (Integrated Development Environment o entorno de desarrollo integrado en español el programa compuesto por un conjunto de herramientas para un programador, es decir, consiste en un editor de código, un compilador, un depurador y en algunos casos un constructor de interfaz gráfica GUI. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios de ellos (42).
Un IDE debe tener las siguientes características:

- Multiplataforma.
- Soporte para diversos lenguajes de programación.
- Integración con Sistemas de Control de Versiones.
- Reconocimiento de Sintaxis.
- Extensiones y Componentes para el IDE.
- Integración con Framework populares.

[18] **Common Language Runtime:** entorno en tiempo de ejecución de lenguaje común. Es un entorno de ejecución para los códigos de los programas que corren sobre la plataforma Microsoft .NET.

- Depurador.
- Importar y Exportar proyectos.
- Múltiples idiomas.
- Manual de Usuarios y Ayuda (43).

- **Visual Studio**

Microsoft Visual Studio es un entorno de desarrollo integrado (*IDE*, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación, tales como C++, C#, J#, y Visual Basic .NET, al igual que entornos de desarrollo web como ASP.NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles (44).

Visual Studio está concebido para el trabajo en el entorno .NET y permite explotar, como ningún otro IDE, todas las ventajas de dicha plataforma (45).

- **NetBeans**

NetBeans IDE es un entorno de desarrollo visual de código abierto, libre y gratuito sin restricciones de uso. Es utilizado para aplicaciones programadas mediante Java, aunque puede ser utilizado para programar en otros lenguajes (46). Dentro de sus características podemos encontrar también, que es multiplataforma y con él es posible desarrollar desde aplicaciones para la Web, para dispositivos portátiles, como móviles o Pocket PC, hasta potentes aplicaciones de Escritorio.

1.6 Propuesta de herramientas y tecnologías a utilizar

Como herramientas y tecnologías a utilizar para el desarrollo del componente se propone que:

- El modelado del software se realice usando el lenguaje UML para especificar, construir y definir de forma gráfica y documental el diseño de la solución, proporcionándole un soporte concreto a la metodología seleccionada.
- La herramienta CASE a utilizar para el proceso de modelado sea Visual Paradigm Enterprise Edition en su versión 8.0 ya que esta constituye un software de alta eficiencia que permite realizar ingeniería tanto inversa como directa, además de permitir la generación de documentación de forma automática en diferentes formatos.
- El lenguaje de programación sea C# por ser orientado a objetos y de un alto grado de efectividad.

- El lenguaje seleccionado sea puesto en práctica a través del IDE Visual Studio por representar la herramienta más factible para el trabajo con este lenguaje y permitir el desarrollo de aplicaciones de alta calidad, robustez y seguridad.

Conclusiones parciales

El análisis realizado sobre los diferentes aspectos y conceptos involucrados en el proceso de medición de calidad de imágenes de huellas dactilares posibilitó un mejor entendimiento del contexto de la investigación y de la problemática a resolver. El estudio de las características y funcionamiento de algunos de los mecanismos utilizados por algunos de los sistemas biométricos existentes a nivel mundial, así como las soluciones existentes en el país, demostraron la necesidad de que se desarrolle un módulo de medición de calidad de imágenes de huellas. Como resultado del análisis de las peculiaridades de las metodologías de desarrollo, herramientas y tecnologías existentes se seleccionaron aquellas que apoyarán el ciclo de vida de la solución.

Capítulo II: Características del componente

Introducción

En el presente capítulo se exponen las principales características de la solución informática a desarrollar. Se realiza la captura de los requisitos funcionales y no funcionales, se conforman las historias de usuario correspondientes a estos. Se define el modelo de dominio de la aplicación con el objetivo de abarcar los principales conceptos con los que trabaja el componente. Se puntualizan el plan de entregas, el de iteraciones y las tareas ingenieriles como parte de la etapa de planificación del software. Se describe la arquitectura del módulo, así como los patrones de diseño a aplicar y se exponen el diagrama de clases del diseño y las tarjetas CRC asociadas a cada una de las clases definidas en él.

2.1 Descripción del problema

La calidad de la imagen de una huella dactilar incide directamente en la complejidad de los algoritmos de extracción de minucias que se requieran, lo que repercute en la eficiencia y rapidez de un sistema biométrico de huellas dactilares. Si la imagen capturada es de mala calidad, el algoritmo de extracción de características puede extraer un conjunto de minucias que en realidad no existan, o no tengan valor identificativo, dando paso al error denominado FTP. En la calidad de las imágenes de huellas dactilares influyen tanto el sensor utilizado para su captura como la presión ejercida por el dedo y las condiciones de la piel.

En el centro CISED se está desarrollando un componente de extracción de características de huellas dactilares. Con el objetivo de aumentar la calidad del proceso se hace necesaria la implementación de una estrategia que garantice que imágenes de baja calidad sean excluidas del proceso de extracción.

2.2 Propuesta de solución

El presente trabajo propone un componente de medición de calidad de imágenes de huellas dactilares que, utilizado por un sistema de enrolamiento, permitirá la selección de imágenes adecuadas para el proceso de extracción de características. El componente evaluará la calidad de la imagen a través del método de Chaohong-Tulyakov-Govindaraju, el cual ofrece una clasificación de calidad después de analizar la imagen de la huella de manera local y global.

2.3 Modelo de dominio

El modelo de dominio, o modelo conceptual, se define con el objetivo de garantizar la comprensión del entorno del componente, mostrando los principales conceptos o entidades que intervienen en el negocio. Se definen como conceptos fundamentales:

- **Imagen_Huella_Dactilar**: imagen que contiene la huella dactilar a medir su calidad.

- **Algoritmo_Medición_Calidad**: algoritmo de medición de calidad de imágenes de huellas dactilares. Puede basarse en características locales, globales o en clasificadores.
- **Medida_Calidad_Algoritmo**: resultado de un algoritmo de medición de calidad de imágenes de huellas dactilares. Es una medida cuantitativa que se obtiene como resultado de la aplicación de este tipo de algoritmo a una imagen de huella dactilar.
- **Calidad_Imagen_HD**: medida cualitativa que responde a la combinación de los resultados generados por los algoritmos de medición de la calidad de imágenes de huellas dactilares. Responde a la calidad general de la imagen de la huella como resultado de combinar las medidas de calidad independientes de cada algoritmo aplicado.

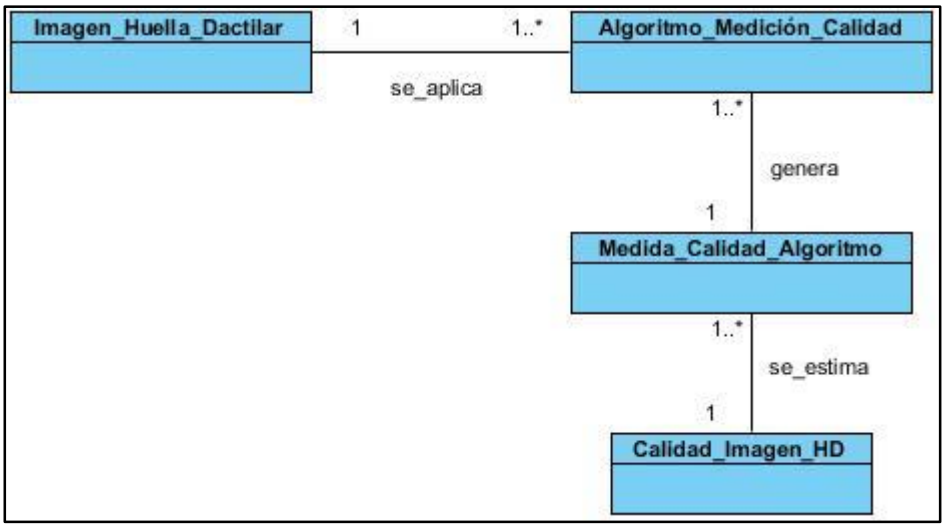


Figura 9 Modelo de dominio.

2.4 Captura de requisitos

La captura de requisitos es uno de los pasos cruciales en el desarrollo de cualquier software. Esta tarea está encaminada a identificar lo que el cliente quiere, analizar las necesidades y especificar los requerimientos de la solución sin ambigüedades.

Los *requisitos funcionales* describen las funciones que el software va a ejecutar, se conocen también como capacidades. Los *requisitos no funcionales* son los que actúan para obligar la solución. Los requisitos no funcionales se conocen a veces como apremios o requisitos de calidad (47).

2.4.1 Principales funcionalidades (RF)

RF-1 Determinar la calidad de una imagen de huella dactilar.

a) Procesar una imagen de huella dactilar.
b) Medir la calidad de una imagen de huella dactilar según la combinación de características locales y globales.

RF-2 Generar el mapa de calidad de una imagen de huella dactilar.

RF-3 Determinar la calidad de un dataset de huellas dactilares (carpeta con un conjunto de imágenes de huellas).

RF-4 Generar mapas de calidad de un dataset de huellas dactilares.

2.4.2 Requerimientos no funcionales (RNF)

Software:

RNF-1. El Sistema Operativo que debe estar instalado en la estación de trabajo en que puede ejecutarse el componente es Windows.

RNF-2. Framework de desarrollo .NET 4.0.

Hardware:

RNF-3. Procesador Intel Pentium 4 o superior.

RNF-4. 1 GB o más de memoria RAM.

RNF-5. CPU 3 GHZ o superior.

Restricciones en el Diseño y en la Implementación:

RNF-6. La implementación del componente debe ser desarrollada en el lenguaje C#.

2.5 Historias de usuario (HU)

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación (23). La redacción de las mismas se hace bajo la terminología del cliente, de forma que sea sencilla, clara y no profundice en los detalles. A continuación se muestran las historias de usuario de mayor relevancia, el resto está definido en los anexos. (**Ver Anexos_5**)

Historia de Usuario	
Número: HU_1	**Usuario:** Sistema.
Nombre de historia: Determinar la calidad de una imagen de huella dactilar.	
Prioridad en negocio: Muy Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 9	**Iteración asignada:** 1 y 2
Programador responsable: Alexei Alayo Rondón	

Descripción: El componente debe medir la calidad de la imagen de la huella dactilar. Para ello debe: • Obtener la matriz y el arreglo de bytes de la imagen. • Procesar los bloques de la imagen y determinar, de cada uno, la media, desviación estándar, uniformidad, suavidad, índice de no homogeneidad y la orientación del campo direccional. • Determinar un índice de calidad global a partir de la concentración de la energía en el espectro de Fourier. • Determinar la calidad de la imagen a través de un criterio basado en la concentración de energía en el espectro de Fourier y el análisis de las características de los bloques
Observaciones: Se debe contar con la imagen de la huella dactilar.

Tabla 5 HU Determinar la calidad de una imagen de huella dactilar.

Historia de Usuario	
Número: HU_2	**Usuario:** Alexei Alayo Rondón
Nombre de historia: Generar el mapa de calidad de una imagen de huella dactilar.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Alexei Alayo Rondón	
Descripción: El componente debe mapear gráficamente la calidad de una imagen de huella dactilar, permitiendo la distinción de zonas de un nivel de calidad específica.	
Observaciones: Se debe disponer de la imagen de la huella dactilar y se debe haber determinado previamente su calidad.	

Tabla 6 HU Generar el mapa de calidad de una imagen de huella dactilar.

2.6 Planificación

2.6.1 Plan de entregas

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente (23).

Guiados por XP, el equipo de desarrollo debe mantener un registro de "velocidad" de desarrollo. La velocidad es establecida en puntos por iteración y está basada principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. La velocidad del proyecto es utilizada para establecer la cantidad de historias que pueden implementarse antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias.

No. de iteración	Fin de la iteración
1	Febrero de 2014
2	Marzo de 2014
3	Abril de 2014

Tabla 7 Plan de entregas por iteraciones.

2.6.2 Plan de iteraciones

El ciclo de desarrollo de software establecido por XP tiene un carácter iterativo-incremental, por lo que deben ser definidas las iteraciones a realizar. Para cada iteración se define el conjunto de HU a implementar.

Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores (23).

Iteración	Historia de Usuario	Duración estimada (semanas)
1	Medición de la calidad de una imagen de huella dactilar.	5
2	Medición de la calidad de una imagen de huella dactilar.	5
	Generar mapa de calidad de una imagen de huella dactilar.	
3	Medición de calidad de un set.	3
	Generar mapa de calidad de un set.	

Tabla 8 Plan de iteraciones.

En la primera iteración se priorizará la primera historia de usuario, de la cual depende el resto del funcionamiento del componente. Por la complejidad y cantidad de subprocesos que requiere la determinación de la calidad de una imagen de huella dactilar, se comienza la implementación de este proceso en la primera iteración, y se prolonga hasta la segunda.

2.6.3 Tareas ingenieriles

Las historias de usuario son descompuestas en tareas de programación (task card) y asignadas a los programadores para ser implementadas durante una iteración (48). Las tareas de programación a realizarse en cada iteración son definidas en la siguiente tabla. En los anexos se exponen las descripciones de cada una de dichas tareas. (**Ver Anexo_6**)

Iteración	Historia de Usuario	Tarea de programación
1	Determinar la calidad de una imagen de huella dactilar dada.	• Obtener el histograma de intensidades. • Calcular características locales de un bloque de una imagen. • Obtener el espectro de Fourier. • Determinar la coherencia local del campo de orientación de los bloques.
2	Determinar la calidad de una imagen de huella dactilar dada.	• Calificar los bloques de una imagen según las características locales de la misma. • Determinar el factor de calidad de global de una imagen a partir de la concentración de energía en el espectro. • Determinar la calidad general de una imagen de huella dactilar.
	Generar mapa de calidad de una imagen de huella dactilar.	• Obtener el mapa de calidad de una imagen.
3	Determinar la calidad de un dataset.	• Determinar la calidad de cada una de las imágenes de huellas dactilares de un dataset.
	Generar mapa de calidad de un dataset.	• Obtener el mapa de calidad de las imágenes que conforman un dataset.

Tabla 9 Distribución de tareas ingenieriles por iteración.

2.7 Diseño

El papel del diseño en el ciclo de vida de un software es facilitar la comprensión de su funcionamiento y proveer una representación o modelo del mismo con el propósito de definirlo con los suficientes detalles como para permitir su realización física. El modelo de diseño provee una representación arquitectónica del software que sirva de punto de partida para las tareas de implementación, dando al traste con los requisitos del sistema.

2.7.1 Descripción de la arquitectura

La IEEE define la arquitectura de software como la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución (49). Según Roger Pressman: "*En su forma más simple, la arquitectura del software es la estructura u organización de los componentes del programa, la manera en que estos interactúan y la estructura de datos que utilizan*" (24).

O sea, la arquitectura de software es una forma de representar sistemas mediante el uso de la abstracción, de forma que aporte el más alto nivel de comprensión de los mismos. Esta representación incluye los componentes fundamentales del software, su comportamiento y formas de interacción para satisfacer los requisitos del sistema.

Para el presente trabajo de diploma se propone una arquitectura en capas.

Un sistema o arquitectura en capas representa una estructura organizada de forma jerárquica de modo que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones de la capa inmediatamente inferior (50).

La arquitectura en capas se basa en la asignación jerárquica de roles y responsabilidades con el objetivo de separar los diferentes aspectos del desarrollo. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada (funcionalidades asociadas a las interfaces de usuario, procesamiento de reglas de negocios o acceso a datos). Esta arquitectura describe la descomposición de servicios de forma que la mayoría de la interacción ocurre solamente entre capas vecinas. Una vista en capas permite documentar típicamente los límites de abstracción entre las partes, estableciendo las formas en que unas partes pueden interactuar con otras. En la práctica, las capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas.

La arquitectura propuesta para el desarrollo del componente de medición de calidad de imágenes de huellas dactilares identifica como capas:

- **Capa Presentación:** representa la interfaz gráfica de prueba para la interacción con el módulo. A partir de varias opciones, muestra los mapas de calidad de una huella como resultado de la aplicación de un determinado algoritmo de medición de calidad de imagen de huellas dactilares.
- **Capa de Negocio:** refleja la lógica del sistema representando la organización de las clases y relaciones entre estas.

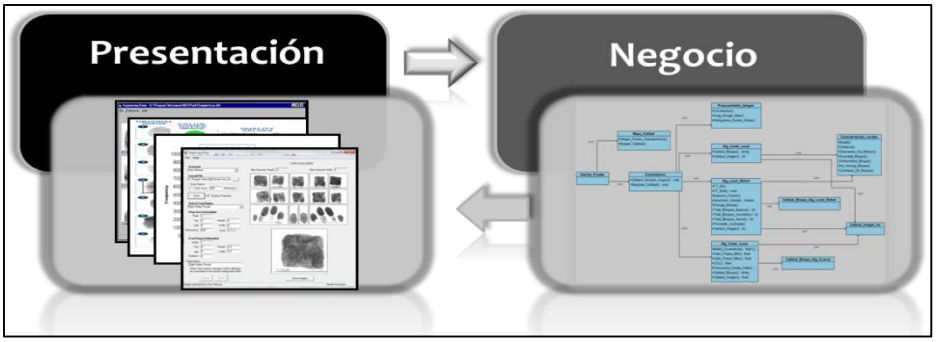


Figura 10 Arquitectura del componente.

2.7.2 Patrones de diseño

Un patrón de diseño es una descripción de la comunicación entre objetos y clases, personalizada para resolver un problema de diseño general en un contexto particular. Identifica clases, instancias, roles, colaboraciones y la distribución de responsabilidades (51). Se presentan como pares de problema-solución con nombre, sugiriendo aspectos relacionados con la asignación de responsabilidades.

Los patrones de diseño se caracterizan por:

- Representar soluciones técnicas a problemas concretos.
- Propiciar la reutilización.
- Representar problemas frecuentes.

Los patrones **GRASP** (Patrones Generales de Software para Asignar Responsabilidades) describen los principios fundamentales de la asignación de responsabilidades a objetos (52). El nombre se eligió para indicar la importancia de captar (*grasping*) estos principios, si se quiere diseñar un software de manera eficaz. En el diseño de la solución propuesta se utilizaron los patrones GRASP experto, bajo acoplamiento y alta cohesión.

Patrón Experto: se usa más que cualquier otro al asignar responsabilidades, es un principio básico que suele utilizarse en el diseño orientado a objetos (52). Consiste en la asignación de una responsabilidad a la clase que cuenta con la información necesaria para llevarla a cabo. El uso de este patrón da pie a un bajo acoplamiento y una alta cohesión, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. El cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos.

En el componente de medición de calidad de huellas dactilares este patrón se evidencia en las clases ***ChaohongWuAlgorithm***, ***LocalQualityAnalysis*** y ***GlobalQualityAnalysis***, en el momento de determinar la calidad de una imagen de huella dactilar, donde la clase ***ChaohongWuAlgorithm*** es la encargada de llevar a cabo este proceso utilizando la información proveniente de las clase asociadas a los análisis local y global respectivamente.

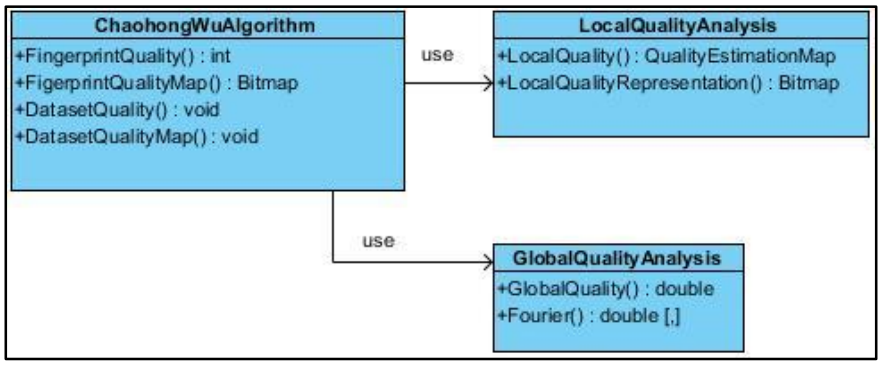


Figura 11 Patrón experto.

Patrón Bajo Acoplamiento: el acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras (52). El bajo acoplamiento soporta el diseño de clases más independientes y reutilizables, lo cual reduce el impacto de los cambios y acrecientan la oportunidad de una mayor productividad. Ejemplo del uso de este patrón en la solución propuesta se muestra en las clases ***LocalQualityAnalysis***, ***ImageBlockQuality*** y ***BlockProcessing***, donde se minimizan las relaciones de estas con el resto de las clases.

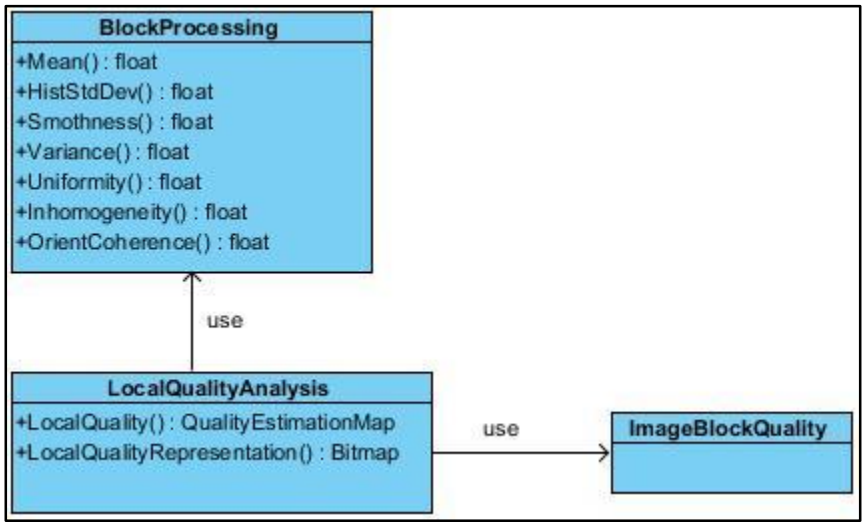


Figura 12 Patrón bajo acoplamiento.

Patrón alta Cohesión: En la perspectiva del diseño orientado a objetos, la cohesión (o más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase (52). Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. En el diseño de la solución que se propone, se evidencia este patrón en la relación que se establece entre las clases ***LocalQualityAnalysis*** y ***BlockProcessing*** al realizar el análisis local de una imagen de huella dactilar.

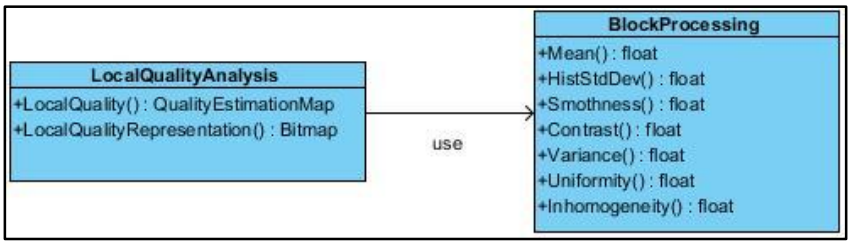


Figura 13 Patrón alta cohesión.

2.7.3 Diagrama de clases del diseño

Los diagramas de clases son los más comunes en el modelado de sistemas orientados a objetos. Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Se usa para modelar la vista estática del diseño de un sistema, visualizar, especificar y documentar modelos estructurales, y para construir sistemas ejecutables a través de ingeniería directa e inversa.

La estructura que guiara el desarrollo del componente de medición de calidad de imágenes de huellas dactilares tendrá como base el siguiente diagrama de clases del diseño.

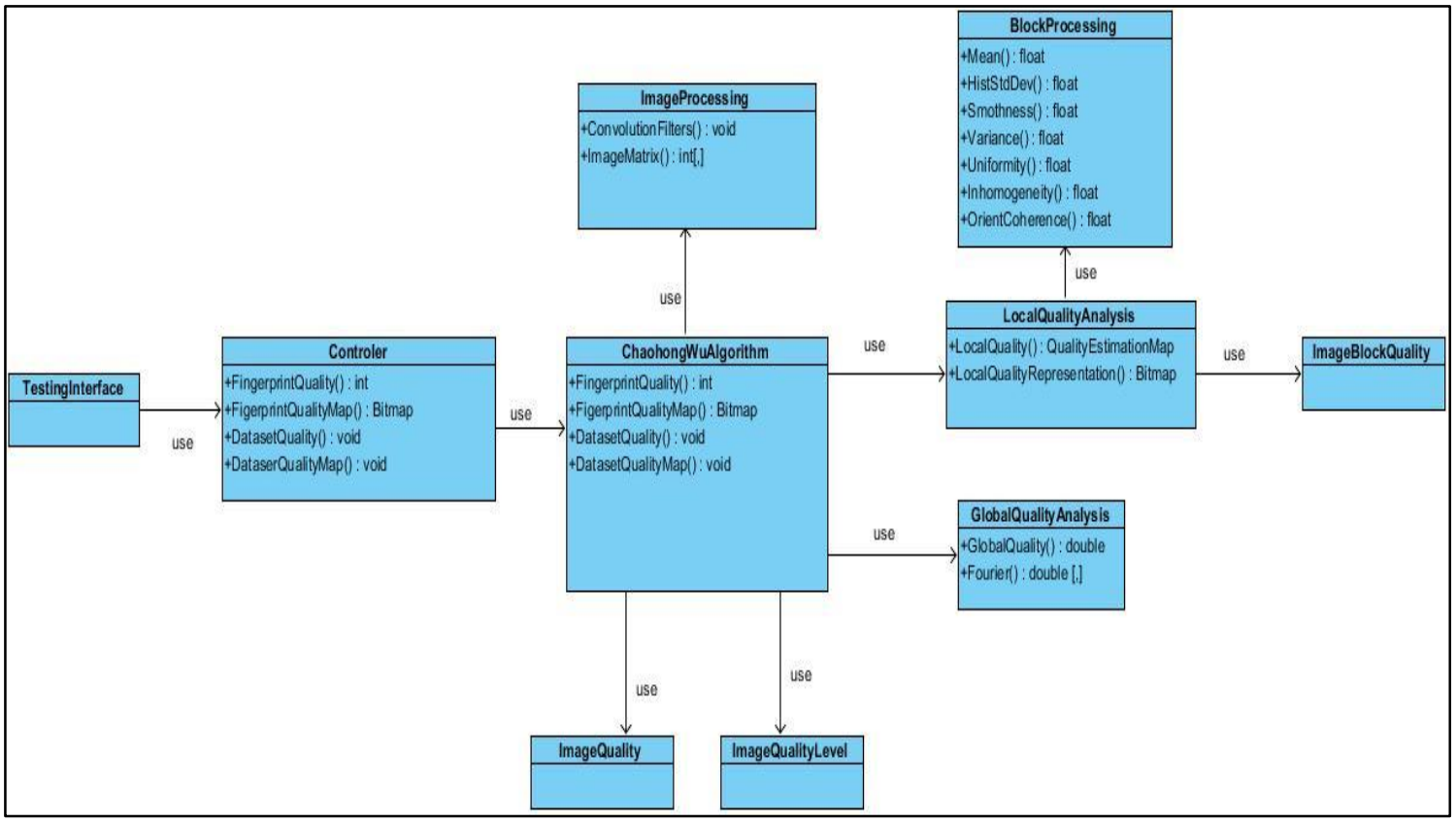


Figura 14 Diagrama de clases.

2.7.4 Tarjetas CRC

Las tarjetas CRC (Clase-Responsabilidad-Colaboración) constituyen uno de los artefactos generados a partir del uso de la metodología XP como guía del desarrollo del componente. Se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores.

Una clase representa cualquier persona, cosa, evento, concepto, pantalla o reporte. Las responsabilidades de una clase son las funcionalidades a su cargo y sus atributos. Los colaboradores de una clase constituyen el resto de las clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades. A continuación se muestran las tarjetas CRC correspondientes las clases ***ChaohongWuAlgorithm***, ***GlobalQualityAnalysis*** y ***LocalQualityAnalysis***, el resto se muestra en los anexos. (**Ver Anexo_7**)

Clase ChaohongWuAlgorithm	
Responsabilidades	**Colaboradores**
Determinar la calidad de una imagen de huella dactilar.	• LocalQualityAnalysis • GlobalQualityAnalysis • MathNet.Iridium • ImageProcessing
Obtener el mapa de calidad de una imagen de huella dactilar.	• LocalQualityAnalysis

Tabla 10 Tarjeta CRC ChaohongWuAlgorithm

Clase LocalQualityAnalysis	
Responsabilidades	**Colaboradores**
Determinar la calidad de los bloques de la imagen.	• BlockProcessing • ImageBlockQuality
Obtener el mapa de calidad local de la imagen.	----------

Tabla 11 Tarjeta CRC LocalQualityAnalysis

Clase GlobalQualityAnalysis	
Responsabilidades	**Colaboradores**
Determinar el factor de concentración de la energía en el espectro de Fourier.	• MathNet.Iridium
Determinar el espectro de Fourier asociado a una imagen.	• MathNet.Iridium

Tabla 12 Tarjeta CRC GlobalQualityAnalysis

Conclusiones parciales

La definición del modelo de dominio permitió identificar con mayor facilidad los requerimientos funcionales del componente, propiciando así la comprensión de su funcionamiento. Las historias de usuario, correspondientes

a cada uno de estos requerimientos, fueron especificadas sin emplear lenguaje técnico y detallando entre otros elementos el tiempo que requieren para su codificación, favoreciendo la estimación del plan de entregas del producto y delimitando el propósito y duración de cada iteración.

Con el uso de la arquitectura en capas especificada y algunos de los patrones GRASP, se logró una mejor organización de la solución, de manera que las clases identificadas y sus relaciones sean las bases para su implementación.

Capítulo III: Implementación y prueba

Introducción

En el presente capítulo se exponen las especificaciones asociadas a la implementación del componente. Se describen los estándares de codificación y el diagrama de despliegue, así como la interfaz de usuario diseñada para probar el correcto funcionamiento del componente. El componente es sometido a pruebas con el objetivo de validar el cumplimiento de los requerimientos del sistema.

3.1 Estrategia de medición del componente

En la sección 1.4.2 es detallado el método de Chaohong-Tulyakov-Govindaraju bajo el título "***Método basado en características locales y globales***". Sin embargo, como resultado de la constante experimentación a la que fue sometido el componente, el método estuvo sujeto a las siguientes modificaciones:

- En el análisis local de las imágenes, se sustituyó el contraste direccional por el chequeo de la coherencia del campo de orientación. Para ello se utilizó el procedimiento del algoritmo basado en la coherencia local del campo de orientación definido en la sección 1.4.2 (**Ver Definición del Procedimiento**).
- Se adicionan dos clasificaciones para los bloques: corruptos y background. La primera clasificación corresponde a los bloques de la huella que no tienen bien definida la estructura cresta-valle, mientras que la segunda se asocia a los bloques de la imagen que no forman parte de la huella, es decir, los bloques del fondo. A continuación se definen los umbrales asociados a la clasificación de los bloques:
  - Un bloque se califica de "corrupto" si su media es menor que 100, o su coherencia del campo de orientación esta entre 0 y 0.25.
  - Si la coherencia del campo de orientación del bloque es nula, entonces se clasifica como "background".
  - Un bloque es "bueno" si su coherencia del campo de orientación mayor o igual que 0.75.
  - Si la coherencia del campo de orientación esta entre 0.50 y 0.75 se clasifica el bloque como "normal".
  - Se establece que siempre que la coherencia del campo de orientación del bloque esté entre 0.25 y 0.50:
    - Si el índice de no homogeneidad de un bloque es mayor que 3000 o su media es mayor que 160 entonces se califica como "seco". Recibe la misma calificación si el cociente entre la media y la coherencia del campo de orientación es mayor que 5, y el cociente entre la uniformidad y la suavidad del bloque es mayor que 20.
    - Se clasifica un bloque como "húmedo" si su índice de no homogeneidad está entre 500 y 1500. Recibe la misma calificación si su media es menor que 60 y su desviación estándar menor que 1300.
- Para determinar la calidad global de una imagen, se analizó el espectro de Fourier entre las frecuencias 30 y 60, calculando el pico dominante entre estas frecuencias en 5 direcciones $(0, \pi, \frac{\pi}{2}, \frac{\pi}{4}, \frac{3\pi}{4})$ (ver figura 15). A

continuación se promediaron dichos valores, generando un índice de cuan fuerte se concentra la energía en el espectro. La constante prueba de este procedimiento, con múltiples imágenes de huellas dactilares, favoreció el establecimiento de umbrales adecuados para la diferenciación de la calidad de las mismas. Los umbrales son resumidos en la tabla 14.

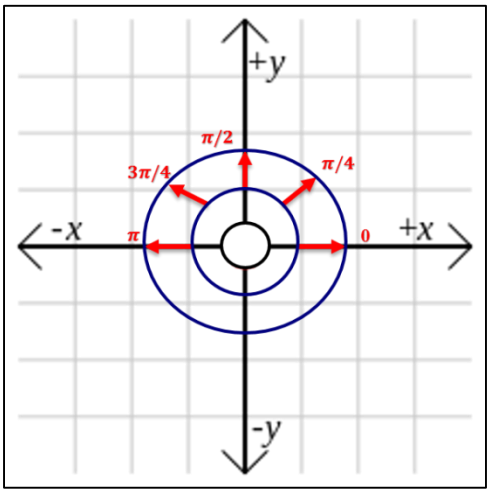


Figura 15 Direcciones analizadas en la región en forma de anillo del espectro de Fourier.

Factor de calidad global (índice de concentración de energía)	Calidad de la imagen de huella dactilar
Mayor que 5	Buena
Entre 4 y 5	Regular
Menor que 4	Mala (imágenes muy claras)

Tabla 13 Umbrales de calidad según el análisis del espectro.

3.1.1 Métricas para determinar la calidad de la imagen

Como consecuencia de las adaptaciones aplicadas al método fueron ajustadas las métricas de clasificación de la calidad. En los anexos se muestra el código del método ***FingerprintQuality***, el cual implementa las métricas definidas para determinar la calidad de una imagen (**Ver Anexo 8**). A continuación se muestran algunos de los parámetros de clasificación:

- Si el porcentaje de bloques buenos es mayor o igual que 75% y el índice de calidad global es mayor o igual que 5.0, la imagen se clasifica como “buena” (Nivel 1).
- Si el porcentaje de bloques buenos es menor que 75% y el índice de calidad global es mayor o igual que 5.0:
    - La calidad es “buena” si el porcentaje de bloques buenos supera el 65%.
    - La imagen es de calidad “normal” (nivel 2) si el porcentaje de bloques buenos es mayor que 50%.

  - Se clasifica la imagen de “manchada/húmeda” (nivel 3) si el porcentaje de bloques buenos es mayor que 30%.
- Si el porcentaje de bloques buenos es mayor o igual que 75% y el índice de calidad global es menor que 5.0:
  - La imagen se clasifica de “normal” si el índice de calidad global es mayor o igual que 4.0.
  - Si el índice de calidad global es mayor o igual que 4.5, se clasifica la imagen como “buena”.
  - La imagen se clasifica como “seca” (nivel 4) en cualquier otro caso.
- Si el porcentaje de bloques buenos está entre 50% y 75%, y el índice de calidad global está entre 4.0 y 5.0, la calidad de la imagen se define como “normal”.
- Si el índice de calidad global es menor que 4.0, el porcentaje de bloques buenos es menor o igual que 30% y el de los bloques secos supera el 30%, la imagen se clasifica como “seca”.
- Si el índice de calidad global es menor que 4.0, el porcentaje de bloques buenos es menor o igual que 30% y el de los bloques húmedos supera el 30%, la imagen se clasifica como “manchada/húmeda”.
- La imagen se clasifica como “estropeada” (nivel 5) si su porcentaje de bloques corruptos supera el 30%.

3.2 Implementación

Una vez definidas las historias de usuario y concluido el diseño del componente, tiene lugar la codificación de la solución propuesta, cuyos objetivos están encaminados a desarrollar de forma iterativa e incremental un producto completo listo para el despliegue, obteniendo versiones útiles de forma rápida, las que paulatinamente completan el desarrollo del componente.

3.2.1 Estándares de codificación

Entre las prácticas propuestas por XP para el desarrollo de software se encuentra la posesión colectiva del código, la cual garantiza que cualquier miembro del equipo de desarrolladores pueda modificar cualquier parte del sistema. Con el objetivo de complementar esta práctica la metodología establece el uso de estándares de codificación, de forma tal que el código sea estructurado siguiendo un criterio único.

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurar que todos los programadores del proyecto trabajen de forma coordinada (53). Las técnicas de codificación incorporan muchos aspectos del desarrollo del software. Aunque generalmente no afectan a la funcionalidad de la aplicación, sí contribuyen a una mejor compresión del código fuente (54).

En la propuesta de solución, para declarar el nombre de las variables, métodos y clases se tendrán en cuenta las siguientes convenciones:

- Se utilizarán nombres descriptivos y sugerentes que contribuyan a una mejor comprensión del código.
- Para la nomenclatura de las clases y los métodos se utilizará el estilo de capitalización Pascal, con el cual se capitaliza la primera letra de cada palabra. Ejemplo ProcesamientoImagen.
- Los nombres de las variables responderán al estilo de capitalización Camel, con el cual se capitaliza la primera letra de las palabras (excepto la primera palabra). Ejemplo calidadGeneral.
- No usar nombres de variables que coincidan con palabras reservadas.
- Los comentarios deben estar en el mismo nivel del código.
- No escribir comentarios para cada línea de código o para cada variable declarada.
- Las llaves se deben poner al mismo nivel del código que las contiene.
- Debe dejarse una y solo una línea en blanco entre cada método dentro de las clases.
- Usar un espacio simple antes y después de cada operador.
- Evitar nombres que difieran en una letra o en el uso de mayúsculas.

3.2.2 Diagrama de componentes

Un componente es una parte modular desplegable y reemplazable de un sistema que encapsula implementación y expone un conjunto de interfaces (24). Usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Un componente puede comprender una gran porción del sistema.

Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. Los diagramas de componentes modelan la vista estática del software y se representan como un grafo de componentes software unidos por medio de relaciones de dependencia. A continuación se muestra el diagrama de componentes definido para la solución propuesta:

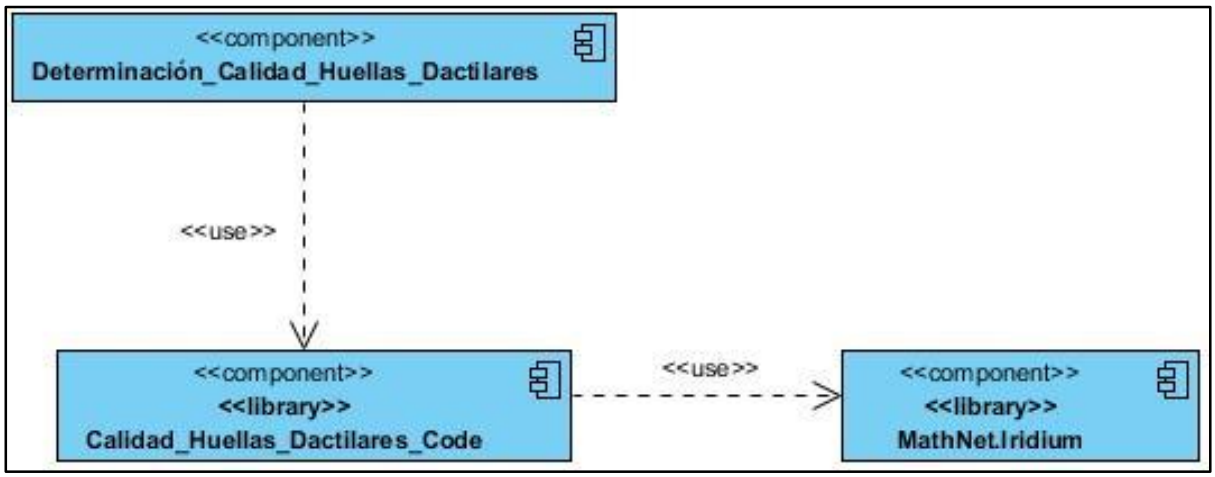


Figura 16 Diagrama de componentes.

- **Library Calidad_Huellas_Dactilares_Code:** librería que incluye todas las clases y relaciones definidas en el diagrama de clases del diseño, a excepción de la clase destinada a la interfaz de prueba.
- **Library MathNet.Iridium:** librería de código abierto que apoya el trabajo con funciones matemáticas.

3.2.3 Diagrama de despliegue

El diagrama de despliegue se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación.

Los diagramas de despliegue se representan a través de un grafo de nodos unidos por conexiones de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria. El modelo de despliegue del componente de medición de calidad de imágenes de huellas dactilares está compuesto por un único nodo, en este caso la computadora donde se pueda ejecutar.

Figura 17 Diagrama de despliegue.

3.2.4 Interfaz de prueba

El objetivo del componente propuesto no requiere una interfaz gráfica propia que permita visualizar todo el proceso de medición de la calidad de imágenes de huellas dactilares, sino que constituya un módulo que pueda sea utilizado por un sistema de enrolamiento biométrico basado en huellas dactilares. Sin embargo, con el fin de visualizar y comprobar el correcto funcionamiento del componente, se definió una interfaz gráfica de prueba. Las interfaces asociadas a cada funcionalidad del componente se muestran en los anexos.

Figura 18 Interfaz que muestra la calidad de una imagen de huella dactilar.

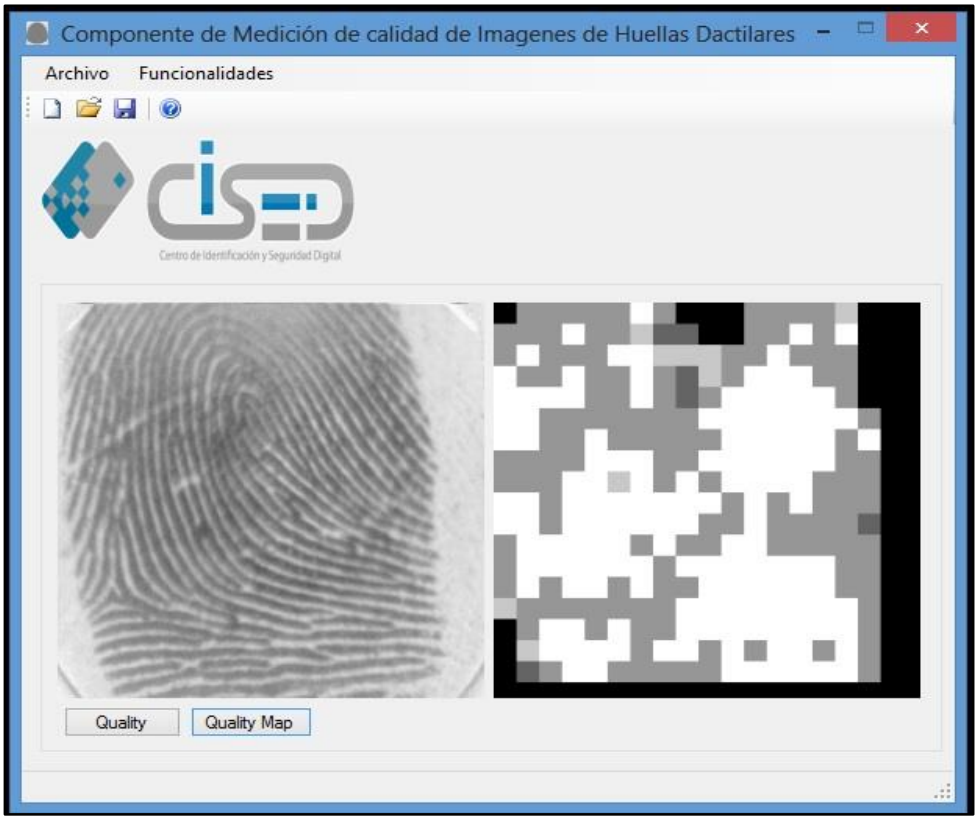


Figura 19 Interfaz que muestra el mapa de calidad de una imagen de huella dactilar.

3.3 Pruebas

La prueba de software se puede definir como una actividad en la cual un sistema o uno de sus componentes se ejecuta en circunstancias previamente especificadas. Los resultados obtenidos se registran para realizar un proceso de evaluación en el que los mismos se comparan con los resultados esperados para localizar fallos en el software. La prueba del software es un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación.

3.3.1 Pruebas de aceptación

Las pruebas de aceptación, también llamadas pruebas funcionales, son supervisadas por el cliente basándose en los requerimientos tomados de las historias de usuario. Cada una de las historias de usuario seleccionadas por el cliente debe tener una o más pruebas de aceptación, de las cuales deberán determinar los casos de prueba e identificar los errores que serán corregidos.

Las pruebas de aceptación son pruebas de caja negra, que representan un resultado esperado de determinada transacción con el sistema. Para que una historia de usuario se considere aprobada, deberá pasar todas las pruebas de aceptación elaboradas para dicha historia (55).

A continuación se muestran los casos de prueba correspondientes a las historias de usuario "*Determinar la calidad de una imagen de huella dactilar*" y "*Generar mapa de calidad de una imagen de huella dactilar*", por representar el proceso fundamental del negocio. Los casos de prueba asociados al resto de las historias de usuario son definidos en los anexos (**Ver Anexo_9**).

Caso de Prueba de Aceptación	
Código: CP1_HU1	**HU_1:** Determinar la calidad de una imagen de huella dactilar.
Responsable: Miriela Velázquez Arias	
Descripción: Prueba de funcionalidad para verificar la medición de calidad de una imagen de huella dactilar.	
Condiciones de Ejecución: El proceso debe recibir una imagen de huella dactilar.	
Entrada / Pasos de ejecución: Luego de cargar una imagen de huella dactilar en la interfaz de prueba, el sistema debe determinar la calidad de la misma automáticamente a través del algoritmo definido.	
Resultado Esperado: Se muestra en la interfaz de prueba el índice de calidad de la imagen de huella dactilar seleccionada y el tiempo requerido para su determinación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 14 Caso de prueba de aceptación 1.

Caso de Prueba de Aceptación	
Código: CP2_HU2	**HU_1:** Generar el mapa de calidad de una imagen de huella dactilar.
Responsable: Miriela Velázquez Arias	
Descripción: Prueba de funcionalidad para verificar la visualización del mapa de calidad asociado a una imagen de huella dactilar.	
Condiciones de Ejecución: El proceso debe recibir una imagen de huella dactilar.	
Entrada / Pasos de ejecución: Luego de cargar una imagen de huella dactilar en la interfaz de prueba y determinar la calidad de la misma, se genera el mapa de calidad correspondiente.	
Resultado Esperado: Se muestra en la interfaz de prueba el mapa de calidad asociado a la huella dactilar seleccionada.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 15 Caso de prueba de aceptación 2.

3.3.2 Pruebas de fiabilidad

Con el objetivo de garantizar la eficiencia del componente desarrollado se realizaron pruebas de fiabilidad, ya que las pruebas propuestas por XP no validan del todo esta variable. Se decidió comparar el comportamiento del componente desarrollado con otra estrategia de medición de calidad de imágenes de huellas dactilares, con el objetivo de analizar tanto la correlación entre ellos como el nivel de certeza del componente.

Herramienta utilizada

Para realizar el estudio se utilizó la distribución NBIS, cuyo código fuente está implementado en el lenguaje C. Del NBIS se utilizó el paquete **MINDTCT**, el cual toma una imagen de huella dactilar y localiza todas las minucias de la imagen, determinando su ubicación, dirección, tipo y calidad.

MINDTCT calcula la calidad de zonas específicas de la imagen, incluyendo el flujo direccional de crestas de la misma, generando además un mapa de calidad a partir de características locales (contraste, dirección y flujo de crestas). Detecta regiones de bajo contraste, bajo flujo de cresta y alta curvatura. Las tres últimas condiciones representan áreas inestables de la imagen donde la detección de minucias es poco confiable, y juntas se usan para definir niveles de calidad de la imagen de 1(mejor calidad) a 5 (peor calidad).

Prestaciones de Hardware

Las pruebas fueron ejecutadas en una PC con las siguientes prestaciones:

- 2GB de RAM.
- Sistema Operativo Windows 7 (32 bits).
- Procesador Intel Celeron a 2.60GHZ.

Bases de Datos

Se utilizaron las bases de datos **DB_A** y **DB_B** de la *Fingerprint Verification Competition* [19](FVC) del año 2000. Cada base de datos incluye cuatro subconjuntos de huellas (DB1, DB2, DB3, DB4) según la tecnología utilizada para su adquisición, los cuales tienen un total de 80 huellas cada uno (8 impresiones por dedo). Sus características se muestran a continuación:

	Tipo de Sensor	Dimensiones de la imagen	Resolución
DB1	Sensor Óptico Bajo-Costo	300x300	500 dpi
DB2	Sensor Capacitivo Bajo-Costo	256x364	500 dpi
DB3	Sensor Óptico	448x478	500 dpi
DB4	Generador Sintético	240x320	Alrededor de 500 dpi

Tabla 16 Bases de datos para las pruebas.

En las figuras 18 y 19 se muestran imágenes de huellas dactilares de las bases de datos utilizadas, donde a), b), c) y d) corresponden a los subconjuntos DB1, DB2, DB3 y DB4 respectivamente.

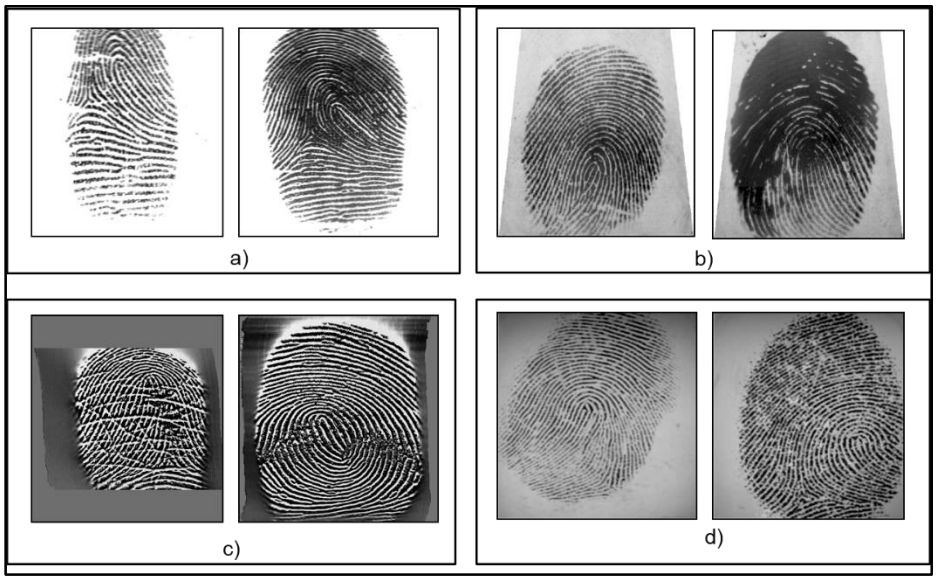


Figura 20 Imágenes de huellas dactilares de DB_A.

[19] **Fingerprint Verification Competition:** Competencia de Verificación de Huellas Dactilares (*FVC* por sus siglas en inglés).Competencia internacional centrada en la evaluación de software de verificación de huellas dactilares. Entre sus organizadores se encuentran las universidades de Madrid, San José y Michigan.

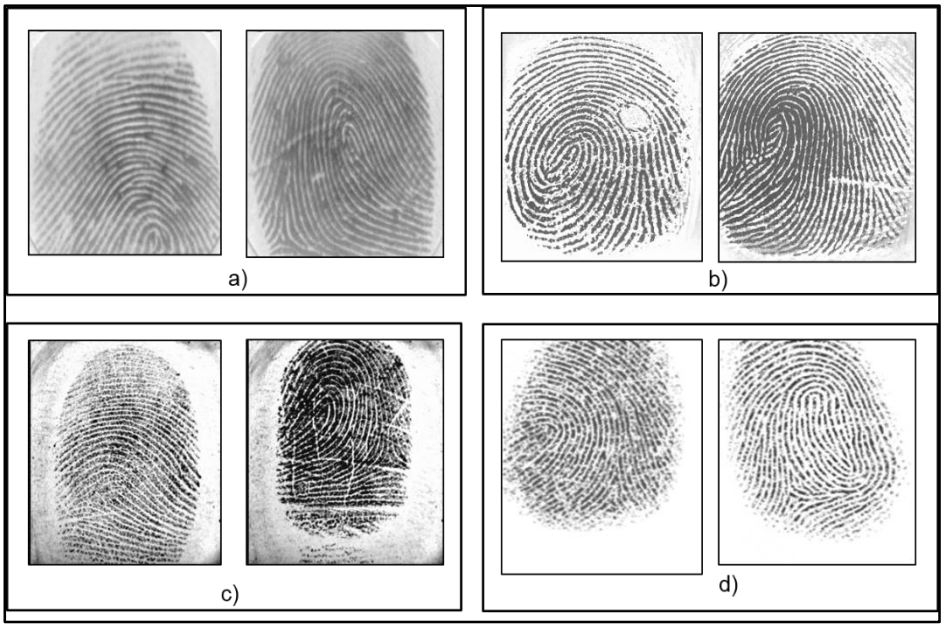


Figura 21 Imágenes de huellas dactilares de DB_B.

Experimentación

Se determinó la calidad de cada una de las imágenes de huellas dactilares de las bases de datos a través del componente desarrollado y del software NBIS. El proceso arrojó datos estadísticos asociados a la cantidad de huellas clasificadas por ambos sistemas para cada nivel de calidad. A partir del uso de las tecnologías actuales fue generado además un factor de correlación entre ambas estrategias de medición de calidad para cada base de datos.

Un índice de correlación establece la relación o dependencia que existe entre dos variables. Se considera que dos variables cuantitativas están correlacionadas cuando los valores de una de ellas varían sistemáticamente con respecto a los valores homónimos de la otra. Se utilizó el coeficiente de correlación de Pearson, el cual se obtiene dividiendo la covarianza de dos variables entre el producto de sus desviaciones estándar.

Fuente: DB_A	Sub-Conjunto: DB1_A			Cantidad de imágenes: 80	
Niveles de calidad	**Nivel 1**	**Nivel 2**	**Nivel 3**	**Nivel 4**	**Nivel 5**
NBIS	43	27	10	0	0
Componente	32	35	0	4	8
Nivel de correlación	0.2832				

Tabla 17 Aplicación de las estrategias de medición sobre DB1_A.

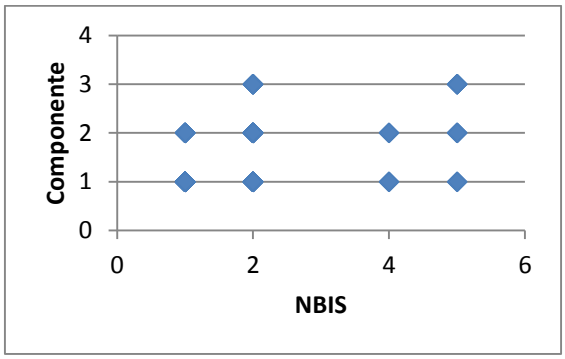


Figura 22 Correlación entre el componente desarrollado y el NBIS en DB1_A.

Fuente: DB_A	Sub-Conjunto: DB2_A			Cantidad de imágenes: 80	
Niveles de calidad	**Nivel 1**	**Nivel 2**	**Nivel 3**	**Nivel 4**	**Nivel 5**
NBIS	9	41	28	0	2
Componente	18	36	0	15	10
Nivel de correlación	0.5311				

Tabla 18 Aplicación de las estrategias de medición sobre DB2_A.

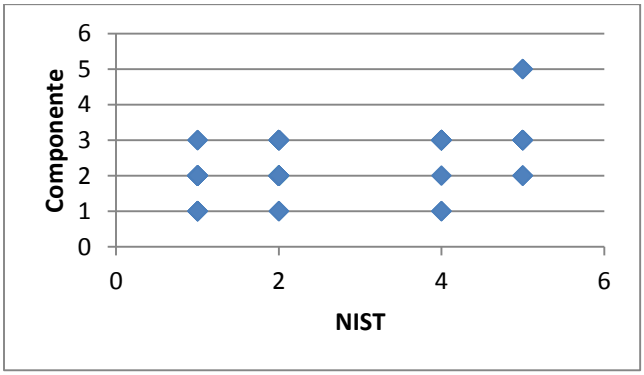


Figura 23 Correlación entre el componente desarrollado y el NBIS en DB2_A.

Fuente: DB_A	Sub-Conjunto: DB3_A			Cantidad de imágenes: 80	
Niveles de calidad	**Nivel 1**	**Nivel 2**	**Nivel 3**	**Nivel 4**	**Nivel 5**
NBIS	55	11	6	8	0
Componente	0	34	0	0	45
Nivel de correspondencia	0.3430				

Tabla 19 Aplicación de las estrategias de medición sobre DB3_A.

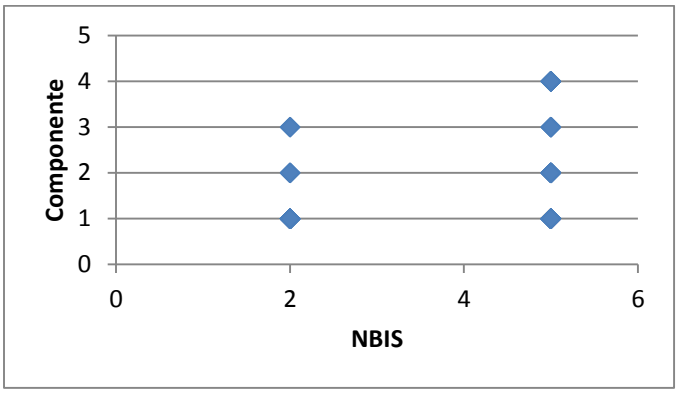


Figura 24 Correlación entre el componente desarrollado y el NBIS en DB3_A.

Fuente: DB_A		Sub-Conjunto: DB4_A		Cantidad de imágenes: 80	
Niveles de calidad	**Nivel 1**	**Nivel 2**	**Nivel 3**	**Nivel 4**	**Nivel 5**
NBIS	47	23	9	0	1
Componente	21	45	2	6	5
Nivel de correspondencia	0.6060				

Tabla 20 Aplicación de las estrategias de medición sobre DB4_A.

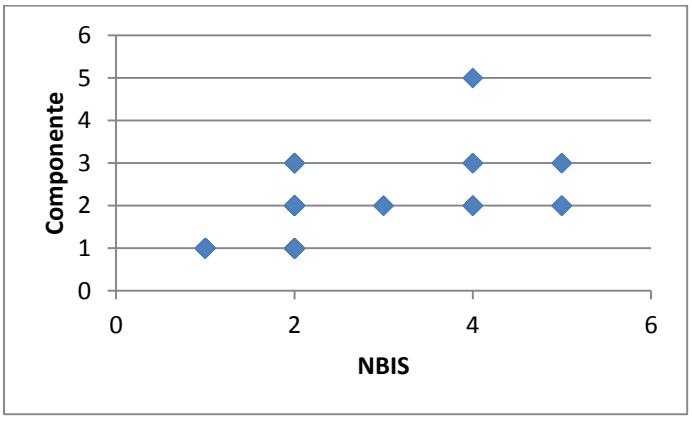


Figura 25 Correlación entre el componente desarrollado y el NBIS en DB4_A.

Fuente: DB_B		Sub-Conjunto: DB1_B		Cantidad de imágenes: 80	
Niveles de calidad	**Nivel 1**	**Nivel 2**	**Nivel 3**	**Nivel 4**	**Nivel 5**
NBIS	21	29	26	0	4
Componente	24	36	0	11	9
Nivel de	0.6204				

correlación	

Tabla 21 Aplicación de las estrategias de medición sobre DB1_B.

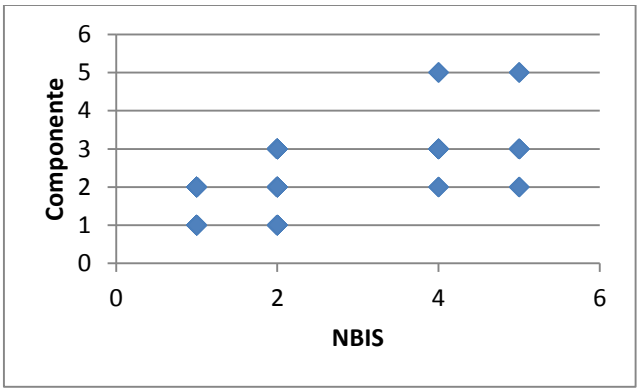


Figura 26 Correlación entre el componente desarrollado y el NBIS en DB1_B.

Fuente: DB_B		**Sub-Conjunto: DB2_B**		**Cantidad de imágenes: 80**	
Niveles de calidad	**Nivel 1**	**Nivel 2**	**Nivel 3**	**Nivel 4**	**Nivel 5**
NBIS	34	31	12	0	3
Componente	15	44	0	4	17
Nivel de correlación	0.4993				

Tabla 22 Aplicación de las estrategias de medición sobre DB2_B.

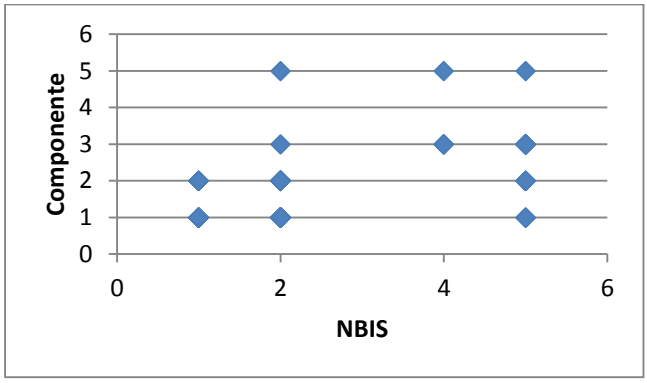


Figura 27 Correlación entre el componente desarrollado y el NBIS en DB2_B.

Fuente: DB_B		**Sub-Conjunto: DB3_B**		**Cantidad de imágenes: 80**	
Niveles de calidad	**Nivel 1**	**Nivel 2**	**Nivel 3**	**Nivel 4**	**Nivel 5**
NBIS	6	18	22	20	14
Componente	12	24	17	12	15
Nivel de correlación	0.4093				

Tabla 23 Aplicación de las estrategias de medición sobre DB3_B.

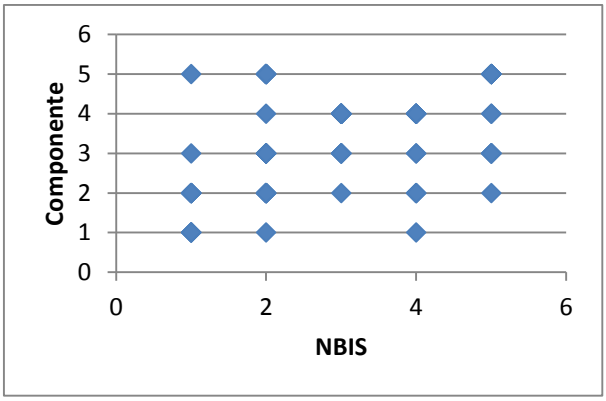


Figura 28 Correlación entre el componente desarrollado y el NBIS en DB3_B.

Fuente: DB_B	Sub-Conjunto: DB4_B			Cantidad de imágenes: 80	
Niveles de calidad	**Nivel 1**	**Nivel 2**	**Nivel 3**	**Nivel 4**	**Nivel 5**
NBIS	7	53	20	0	0
Componente	0	60	1	15	4
Nivel de correlación	0.5133				

Tabla 24 Aplicación de las estrategias de medición sobre DB4_B.

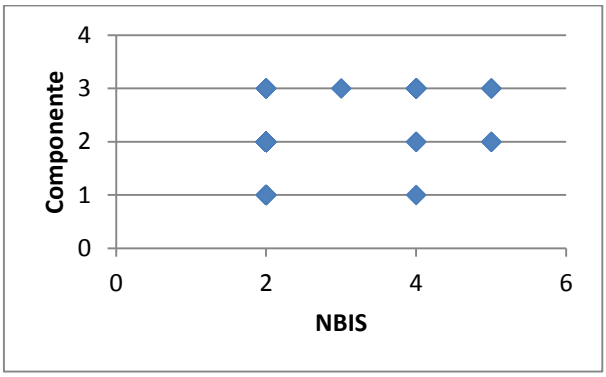


Figura 29 Correlación entre el componente desarrollado y el NBIS en DB4_B.

Análisis de los resultados de la experimentación

La tabla 25 presenta un resumen de los factores de correlación obtenidos tras la aplicación de dos versiones funcionales del componente en la 2ra y 3da iteración respectivamente. Teniendo en cuenta esta tabla y los resultados generados durante la experimentación, se arriba a las siguientes conclusiones:

- La correlación media entre el componente desarrollado y el NBIS se mantuvo estable en el 0.49, lo que significa que ambos sistemas coincidieron en la medición de la calidad de una imagen de huella dactilar en el 50% de los casos.

- El índice de correlación alcanza sus valores máximos, superando el 60%, al ejecutar ambos componentes sobre los datasets DB1_B y DB4_A, y sobre el 50% para los datasets DB2_B, DB4_B y DB2_A. Evidenciando que los sistemas están mejor correlacionados cuando intervienen datasets de imágenes de huellas tomadas con el sensor capacitivo de bajo costo y el generador sintético.
- El componente desarrollado es más riguroso en la medición de la calidad de imágenes de huellas dactilares.

	Correlación 1ra versión	**Correlación 2da versión**
DB1_B	0,558923306	0,620452511
DB2_B	0,472718282	0,499378742
DB3_B	0,391219951	0,409315247
DB4_B	0,577564517	0,513325982
DB1_A	0,211727257	0,382280931
DB2_A	0,610890011	0,531154086
DB3_A	0,489401278	0,343058673
DB4_A	0,5900953	0,606059421
Correlación media	0,487817488	0,488128199

Tabla 25 Índices de correlación para dos versiones funcionales.

3.3.3 Resultados de las pruebas

La aplicación de pruebas de aceptación y de fiabilidad al componente arrojaron los siguientes resultados:

- Las pruebas de aceptación realizadas sobre la versión funcional final del componente generaron un total de 0 no conformidades, garantizando su correspondencia con las especificaciones del cliente.
- El proceso de experimentación al que fue sometido el componente evidenció que solo las imágenes de huellas dactilares de mayor calidad pasen al proceso de extracción de características, garantizando que se disminuya la tasa error FTP.

Conclusiones parciales

El establecimiento de los estándares de codificación permitió desarrollar un código reutilizable y entendible por el equipo de desarrollo. La representación de los diagramas de despliegue y componentes, así como las interfaces de pruebas creadas, facilitaron la visualización de la estructura y el funcionamiento del componente. Las pruebas aplicadas permitieron validar la correspondencia del componente desarrollado a los requerimientos definidos por el cliente.

Conclusiones Generales

Como resultado de la investigación realizada se concluye que:

- El análisis de los elementos teóricos asociados al negocio y el estudio del estado del arte acerca de los procedimientos utilizados en el proceso de medición de calidad de imágenes de huellas dactilares, facilitó la definición de una propuesta de solución acorde a las necesidades existentes.
- El establecimiento de XP como metodología rectora del desarrollo del componente, y el uso de las tecnologías y herramientas seleccionadas, permitieron analizar, describir e implementar los procesos y subprocesos que debían ejecutarse, materializando así una solución acorde a los requerimientos del cliente.
- La implementación del método basado en características locales y globales de las imágenes de las huellas dactilares, así como la definición de métricas, permitió conformar un criterio sólido y abarcador de medición de calidad.
- La implementación del componente de medición de calidad de imágenes de huellas dactilares, luego de ser validada y verificada a partir de las pruebas definidas, y demostrar su correcto funcionamiento, evidenció la satisfacción de las necesidades del cliente. Su uso permitió descartar imágenes de baja calidad del proceso de extracción de características, garantizando en consecuencia la disminución de la tasa del error FTP.
- La aplicación de pruebas experimentales al componente demostró un comportamiento estable al procesar imágenes tomadas con diferentes sensores, asegurando su adaptabilidad y utilidad en ambientes que utilicen distintas tecnologías de captura.

Recomendaciones

Glosario de términos

- **CISED:** Centro de Identificación y Seguridad Digital.
- **Enrolamiento:** proceso mediante el cual se registran individuos en la base de datos de un sistema biométrico.
- **Extracción de características:** proceso mediante el cual se extraen las minucias de una huella dactilar.
- **FBI:** Buró Federal de Investigaciones.
- **IDE:** Entorno Integrado de Desarrollo.
- **Kit de desarrollo:** conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto.
- **MINDTCT:** paquete de la distribución NBIS que toma una imagen de huella dactilar y localiza todas las minucias de la imagen, determinando su ubicación, dirección, tipo y calidad.
- **NBIS (*Nist Biometric Image Software*.):** Distribución de software biométrico desarrollada por el NIST.
- **NFIQ:** algoritmo de medición de calidad de imágenes de huellas dactilares usado para predecir el rendimiento de los algoritmos de comparación.
- **NIST:** Instituto Nacional de Estándares y Tecnologías.
- **SAID:** Sistema Automático de Identificación dactilar.
- **UCI:** Universidad de las Ciencias Informáticas.

Bibliografía Referenciada

1. **Piñol, Marcos J. Lorda.** *Análisis del algoritmo de Maio para la extracción de huellas dactilares. Optimización para una implementación hardware.* España : s.n., 2004.

2Biometrics Security Concerns

3. **Maio, Dario, y otros.** *Handbook of Fingerprint Recognition.* Londres : Springer, 2009. ISBN 978-1-84882-253-5.

4. **Ortega, Victor Hugo García.** *Sistema de Reconocimiento de huellas dactilares para el control de acceso a recintos.* Mexico D.F : s.n., 2006.

5. **Torres, Gualberto Aguilar, Sánchez Pérez, G. y Toscano Medina, K. .** *Fingerprint Recognition Using Local Features and Hu Moments.* Mexico: Instituto Politécnico Nacional : s.n., 2012.

6. **Jain, Anil K. y Feng, Jianjiang.** *FM Model Based Fingerprint Reconstruction from.* Michigan: Department of Computer Science and Engineering Michigan State University : s.n., 2009.

7. **Diez, Juan Carlos Landi.** *Introducción a la biometría informatica y análisis de la huella dactilar como fuente de autenticación en sistemas de seguridad.* Cuenca : s.n., 2007.

8. **Cruz, Aradí Rosales.** Clasificación de huellas digitales mediante minucias. [En línea] 28 de abril de 2009. [Citado el: 10 de noviembre de 2013.] http://ccc.inaoep.mx/~esucar/Clases-mgp/Proyectos/reporte_modelos_huellas.pdf.

9. **Jain, Anil, Chen, Yi y Dass, Sarat.** *Fingerprint Quality Indices for Predicting Authentication Performance.* Michigan: Department of Computer Science and Engineering : s.n.

10. **Fernandez, Fernando Alonso, y otros.** A Comparative Study of Fingerprint Image-Quality Estimation Methods. New York: Institute of Electrical and Electronics Engineers (IEEE) : s.n., 2007. ISSN 1556-6013.

11. National Institute of Standards and Technology (NIST). *NIST Biometric Image Software.* [En línea] 31 de octubre de 2013. [Citado el: 25 de noviembre de 2013.] http://www.nist.gov/itl/iad/ig/nbis.cfm.

12. **Aware.** QualityCheck Software de análisis y calificación de la calidad de la imagen de las huellas dactilares. *Aware.* [En línea] [Citado el: 15 de octubre de 2013.] www.aware.com/es/biometrics/qualitycheck.html.

13. —. AccuScan Escaneado y Digitalización de fichas dactilares con certificación del FBI. *Aware.* [En línea] [Citado el: 15 de noviembre de 2013.] www.aware.com/es/biometrics/accuscan.html.

14. **Ruiz, Reinier Pupo y González, Yerandy Arias.** Serie Científica Universidad de las Ciencias Informáticas. *Componente para determinar la calidad en imágenes de huellas dactilares.* [En línea] 18 de enero de 2012. [Citado el: 8 de noviembre de 2013.] http://publicaciones.uci.cu/index.php/SC/article/view/570. ISSN 2306-2495.

15. **DATYS.** *DATYS Tecnología & Sistemas.* [En línea] [Citado el: 10 de noviembre de 2013.] http://www.datys.cu/docs/Documentación Productos/Biomesys AFIS/Civil/tríptico Biomesys_AFIS.pdf.

16. —. ESPECIFICACIONES TÉCNICAS Biomesys AFIS Civil. *DATYS Tecnología & Sistemas.* [En línea] [Citado el: 15 de noviembre de 2013.] http://www.datys.cu/docs/Documentación Productos/Biomesys AFIS/Civil/Especificaciones Técnicas/Biomesys AFIS Civil V2.0 ET_20.04.10_ E03_Es_D.pdf.

17. **IEEE ICIP.** *Fingerptint quality and validity analysis.* 2002. ISBN: 0-7803-7622-6.

18. **Wu, Chaohong, Tulyakov, Sergey y Govindaraju, Venu.** *Image Quality Measures for Fingerprint Image Enhancement.* Buffalo: Centro para sensores y biométricos unificados : s.n., 2006.

19. *Adaptive flow orientation-based feature extraction in fingerprint images.* **Ratha., Nalini K., Chen, Yi y Jain, Anil K.** 1995, Vol. 28.

20. *Measuring Fingerprint Image Quality Using the Fourier Spectrum.* **al., Ll et.** s.l. : Journal of Electronic Science and Technology of China, 2007, Vol. 5.

21. *Automatic Image Quality Assessment with Application in Biometrics.* **Fronthaler, Hartwig, Kollreider, Klaus y Bigun, Josef.** Suecia : Universidad de Halmstad, 2006. ISBN: 0-7695-2646-2.

22. **Kaisler, Stephen H.** *Software Paradigms.* New Jersey : John Wiley & Sons, 2005. ISBN 0-471-48347-8.

23. **Letelier, Patricio y Penadés, Maria del Carmen.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* España: Universidad Polítca de Valencia : s.n.

24. **Pressman, Roger S.** *Ingenieria de Software. Un enfoque práctico: Sexta Edición.* s.l. : McGraw Hill.

25. **Highsmith, J.** *Agile Software Development Ecosystems.* s.l. : Addison-Wesley, 2002.

26. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *Jacobson, Ivar, Booch, Grady y Rumbaugh, James. El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 2000. ISBN-84-7829-036-2.

27. **Molpeceres, Allberto.** *Procesos de desarrollo: RUP, XP y FDD.* Dresden: T-System ITS GmbH : s.n., 2002.

28. **Kniberg, Henrik.** *Scrum and XP from the Trenches.* s.l. : Crisp, 2007.

29. **Palacios, Juan.** Entorno Virtual de Aprendizaje. *El modelo Scrum.* [En línea] 10 de marzo de 2011. [Citado el: 15 de noviembre de 2013.] http://eva.uci.cu/file.php/161/Documentos/Materiales_complementarios/UD_1_Procesos/Metodologias/SCRUM/El_modelo_SCRUM.pdf.

30. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** *El lenguaje Unificado de Modelado, Manual de Referencia.* s.l. : Addison Wesley, 2000. ISBN: 8478290370.

31. **Muller, Hausi A., Norman, Ronald J. y Slonim, Jacob .** *Computer Aided Software Engineering.* New York : Springer, 2011.

32. **Boggs, Wendy y Boggs, Michael.** *UML with Rational Rose.* New York : Sybex, 2002. ISBN 0−7821−4017−3.

33. Visual Paradigm. *Visual Paradigm.* [En línea] [Citado el: 8 de octubre de 2013.] http://www.visual-paradigm.com/.

34. **Pablo Nicolás Baeza.** *Visual Paradigm DB Visual ARCHITECT SQL.* [En línea] http://www.docstoc.com/docs/96492173/Visual-Paradigm-Studio.

35. **Wilson, Leslie Blacket.** *Comparative Programming Languages .* New York : Addison-Wesley, 1993. ISBN 0-201-56885-3.

36. **Fernández, Ing. Ales Román.** *Diseño e implementación de un software parametrizador del controlador semafórico cubano.* La Habana: Instituto Superior Politécnico José Antonio Echevarría (Departamento de Automática y Computación) : s.n., 2011.

37. **Guéri, Brice-Arnaud.** *Lenguaje C++.* s.l. : ENI, 2005.

38. **Lorente, Carlos del Junco.** *Manual de Programacion C++.* España: Didact S.L : Mad, 2005. ISBN 84-665-4456-9.

39. **Arnold, Ken, Gosling, James y Holmes, David.** *El lenguaje de programación Java-TM.* s.l. : Addison-Wesley Iberoamericana, 2001.

40. **Groussard, Thierry.** *C# 4 Language Bases.* Barcelona : ENI, 2011. ISBN 978-2-7460-6373-1.

41. Introducción al lenguaje C# y .NET Framework. *Microsft.* [En línea] [Citado el: 15 de octubre de 2013.] http://msdn.microsoft.com/es-es/library/aa292164%28v=vs.71%29.aspx.

42. **Salavert, Isidro Ramos y Lozano Pérez, María Dolores.** *Ingeniería del software y bases de datos: tendencias actuales.* España : Ediciones de la universidad de Castilla-La Mancha, 2000. ISBN 84-8427-077-7.

43. **Deitel, Harvey M. y Deite, Paul J.** *Cómo programar en C++.* 2003.

44. **Avery, James.** *Visual Studio Hacks.* s.l. : O'Reilly Media Inc, 2005. ISBN 0-596-00847-3.

45. UTILIZANDO BIBLIOTECAS DE CÓDIGO C EN APLICACIONES C#. *Revista Telem@tica.* [En línea] [Citado el: 2 de 11 de 2013.] revistatelematica.cujae.edu.cu/index.php/tele/article/download/61/60.

46. **Korgent Inc.** *NetBeans 6 in Simple Steps si.* New Delh : Dreamtech Pres, 2008. ISBN 10-81-7722-897-8.

47. **Abran, Alain.** *Software Engineering Body of Knowledge.* 2004. ISBN 0-7695-2330-7.

48. **Calderón, Amaro y Rebaza., Sarah Dámaris Valverde.** *Metodologías Ágiles.* Perú: Universidad Nacional de Trujillo : s.n., 2007.

49. **Hilliard, Rich.** *IEEE-Std-1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems.* 2000.

50. *An Introduction to Software Architecture.* **Garlan, David y Shaw, Mary.** New Jersey : s.n., 1994.

51. *Design Patterns: Elements of Reusable Object-Oriented Software.*

52. **Larman, Craig.** *Uml y Patrones.* México : Addison-Wesley, 1999. ISBN 0-13-748880-7.

53. **Microsoft.** MSDN. *Revisiones de código y estándares de codificación.* [En línea] [Citado el: 3 de abril de 2014.] http://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx.

54. —. MSDN. *Técnicas de codificación.* [En línea] [Citado el: 3 de abril de 2014.] http://msdn.microsoft.com/es-es/library/aa291593(v=vs.71).aspx.

55. **Tobón, Luis Miguel Echeverry y Carmona, Luz Elena Delgado.** *Caso Práctico de la Metodología Ágil XP al Desarrollo de Software.* Pereira: Universidad Teconológica de Pereira : s.n., 2007.

Bibliografía Consultada

- **Abran, Alain.** *Software Engineering Body of Knowledge.* 2004. ISBN 0-7695-2330-7.
- AccuScan Escaneado y Digitalización de fichas dactilares con certificación del FBI. *Aware.* [En línea] [Citado el: 15 de noviembre de 2013.] www.aware.com/es/biometrics/accuscan.html.
- *Adaptive flow orientation-based feature extraction in fingerprint images.* **Ratha., Nalini K., Chen, Yi y Jain, Anil K.** 1995, Vol. 28.
- *Automatic Image Quality Assessment with Application in Biometrics.* **Fronthaler, Hartwig, Kollreider, Klaus y Bigun, Josef.** Suecia : Universidad de Halmstad, 2006. ISBN: 0-7695-2646-2.
- **DATYS.** Sistema automatizado de Identificación civil. *DATYS Tecnología & Sistemas.* [En línea] DATYS. [Citado el: 10 de noviembre de 2013.] http://www.datys.cu/wpinfproducto.aspx?22#.
- **Kaisler, Stephen H.** *Software Paradigms.* New Jersey : John Wiley & Sons, 2005. ISBN 0-471-48347-8.
- **Korgent Inc.** *NetBeans 6 in Simple Steps si.* New Delh : Dreamtech Pres, 2008. ISBN 10-81-7722-897-8.
- **Maio, Dario, y otros.** *Handbook of Fingerprint Recognition.* Londres : Springer, 2009. ISBN 978-1-84882-253-5.
- **Martínez, Lic. Reynel Remón y Reyes, Dr. C. Edel García.** Formas de representación y algoritmos de comparación de impresiones palmares. *Centro de Aplicaciones de Tecnologías de Avanzada.* [En línea] febrero de 2010. [Citado el: 15 de octubre de 2013.] http://www.cenatav.co.cu/doc/RTecnicos/RT%20SerieAzul_038web.pdf. ISSN 2072- 6287.
- MSDN. *Técnicas de codificación.* [En línea] [Citado el: 3 de abril de 2014.] http://msdn.microsoft.com/es-es/library/aa291593(v=vs.71).aspx.
- **Wu, Chaohong, Tulyakov, Sergey y Govindaraju, Venu.** *Image Quality Measures for Fingerprint Image Enhancement.* Buffalo: Centro para sensores y biométricos unificados : s.n., 2006.
- *An Introduction to Software Architecture.* **Garlan, David y Shaw, Mary.** New Jersey : s.n., 1994.
- **Arnold, Ken, Gosling, James y Holmes, David.** *El lenguaje de programación Java-TM.* s.l. : Addison-Wesley Iberoamericana, 2001.

- **Arrieta, Angélica González, García Sánchez, Luis Javier y Alonso Romer, Luis .** Gestión y Reconocimiento Óptico de los Puntos Característicos de Imágenes de Huellas Dactilares. [En línea] 2009. [Citado el: 10 de octubre de 2013.] www.lsi.us.es/redmidas/Capitulos/LMD32.pdf.
- **Avery, James.** *Visual Studio Hacks.* s.l. : O'Reilly Media Inc, 2005. ISBN 0-596-00847-3.
- **Aware.** QualityCheck Software de análisis y calificación de la calidad de la imagen de las huellas dactilares. *Aware.* [En línea] [Citado el: 15 de octubre de 2013.] www.aware.com/es/biometrics/qualitycheck.html.
- Biometrics Security Concerns
- **Boggs, Wendy y Boggs, Michael.** *UML with Rational Rose.* New York : Sybex, 2002. ISBN 0−7821−4017−3.
- **Buzón, José Campo.** *Evaluación de conformidad con ISO/IEC 19784-1 para algoritmos de identificación biométrica.* Madrid: Universidad Carlos III de Madrid. Departamento de Tecnología Electrónica : s.n., 2009.
- **Calderón, Amaro y Rebaza., Sarah Dámaris Valverde.** *Metodologías Ágiles.* Perú: Universidad Nacional de Trujillo : s.n., 2007.
- **Cruz, Aradí Rosales.** Clasificación de huellas digitales mediante minucias. [En línea] 28 de abril de 2009. [Citado el: 10 de noviembre de 2013.] http://ccc.inaoep.mx/~esucar/Clases-mgp/Proyectos/reporte_modelos_huellas.pdf.
- **DATYS.** *DATYS Tecnología & Sistemas.* [En línea] [Citado el: 10 de noviembre de 2013.] http://www.datys.cu/docs/Documentación Productos/Biomesys AFIS/Civil/tríptico Biomesys_AFIS.pdf.
- **Deitel, Harvey M. y Deite, Paul J.** *Cómo programar en C++.* 2003.
- *Design Patterns: Elements of Reusable Object-Oriented Software.*
- **Diez, Juan Carlos Landi.** *Introducción a la biometría informatica y análisis de la huella dactilar como fuente de autenticación en sistemas de seguridad.* Cuenca : s.n., 2007.
- ESPECIFICACIONES TÉCNICAS Biomesys AFIS Civil. *DATYS Tecnología & Sistemas.* [En línea] [Citado el: 15 de noviembre de 2013.] http://www.datys.cu/docs/Documentación Productos/Biomesys AFIS/Civil/Especificaciones Técnicas/Biomesys AFIS Civil V2.0 ET_20.04.10_ E03_Es_D.pdf.

- **Fernandez, Fernando Alonso, y otros.** A Comparative Study of Fingerprint Image-Quality Estimation Methods. New York: Institute of Electrical and Electronics Engineers (IEEE) : s.n., 2007. ISSN 1556-6013.
- **Fernández, Ing. Ales Román.** *Diseño e implementación de un software parametrizador del controlador semafórico cubano.* La Habana: Instituto Superior Politécnico José Antonio Echevarría (Departamento de Automática y Computación) : s.n., 2011.
- **García, Juan López.** *Algoritmo para la identificación de personas basado en huellas dactilares.* Departamento de Ingeniería Electrónica : s.n., 2009.
- **Groussard, Thierry.** *C# 4 Language Bases.* Barcelona : ENI, 2011. ISBN 978-2-7460-6373-1.
- **Guéri, Brice-Arnaud.** *Lenguaje C++.* s.l. : ENI, 2005.
- **Highsmith, J.** *Agile Software Development Ecosystems.* s.l. : Addison-Wesley, 2002.
- **Hilliard, Rich.** *IEEE-Std-1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems.* 2000.
- **IEEE ICIP.** *Fingerptint quality and validity analysis.* 2002. ISBN: 0-7803-7622-6.
- Introducción al lenguaje C# y .NET Framework. *Microsft.* [En línea] [Citado el: 15 de octubre de 2013.] http://msdn.microsoft.com/es-es/library/aa292164%28v=vs.71%29.aspx.
- **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *Jacobson, Ivar, Booch, Grady y Rumbaugh, James. El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 2000. ISBN-84-7829-036-2.
- **Jain, Anil K. y Feng, Jianjiang.** *FM Model Based Fingerprint Reconstruction from.* Michigan: Department of Computer Science and Engineering Michigan State University : s.n., 2009.
- **Jain, Anil, Chen, Yi y Dass, Sarat.** *Fingerprint Quality Indices for Predicting Authentication Performance.* Michigan: Department of Computer Science and Engineering : s.n.
- **Kniberg, Henrik.** *Scrum and XP from the Trenches.* s.l. : Crisp, 2007.
- **Larman, Craig.** *Uml y Patrones.* México : Addison-Wesley, 1999. ISBN 0-13-748880-7.
- **Letelier, Patricio y Penadés, Maria del Carmen.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* España: Universidad Política de Valencia : s.n.

- **Lorente, Carlos del Junco.** *Manual de Programacion C++.* España: Didact S.L : Mad, 2005. ISBN 84-665-4456-9.
- *Measuring Fingerprint Image Quality Using the Fourier Spectrum.* **al., LI et.** s.l. : Journal of Electronic Science and Technology of China, 2007, Vol. 5.
- **Microsoft.** MSDN. *Revisiones de código y estándares de codificación.* [En línea] [Citado el: 3 de abril de 2014.] http://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx.
- **Molpeceres, Allberto.** *Procesos de desarrollo: RUP, XP y FDD.* Dresden: T-System ITS GmbH : s.n., 2002.
- **Muller, Hausi A., Norman, Ronald J. y Slonim, Jacob .** *Computer Aided Software Engineering.* New York : Springer, 2011.
- National Institute of Standards and Technology (NIST). *NIST Biometric Image Software.* [En línea] 31 de octubre de 2013. [Citado el: 25 de noviembre de 2013.] http://www.nist.gov/itl/iad/ig/nbis.cfm.
- **Ortega, Victor Hugo García.** *Sistema de Reconocimiento de huellas dactilares para el control de acceso a recintos.* Mexico D.F : s.n., 2006.
- **Pablo Nicolás Baeza.** *Visual Paradigm DB Visual ARCHITECT SQL.* [En línea] http://www.docstoc.com/docs/96492173/Visual-Paradigm-Studio.
- **Palacios, Juan.** Entorno Virtual de Aprendizaje. *El modelo Scrum.* [En línea] 10 de marzo de 2011. [Citado el: 15 de noviembre de 2013.] http://eva.uci.cu/file.php/161/Documentos/Materiales_complementarios/UD_1_Procesos/Metodologias/SCRUM/El_modelo_SCRUM.pdf.
- **Piñol, Marcos J. Lorda.** *Análisis del algoritmo de Maio para la extracción de huellas dactilares. Optimización para una implementación hardware.* España : s.n., 2004.
- **Pressman, Roger S.** *Ingenieria de Software. Un enfoque práctico: Sexta Edición.* s.l. : McGraw Hill.
- **Ruiz, Reinier Pupo y González, Yerandy Arias.** Serie Científica Universidad de las Ciencias Informáticas. *Componente para determinar la calidad en imágenes de huellas dactilares.* [En línea] 18 de enero de 2012. [Citado el: 8 de noviembre de 2013.] http://publicaciones.uci.cu/index.php/SC/article/view/570. ISSN 2306-2495.
- **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** *El lenguaje Unificado de Modelado, Manual de*

Referencia. s.l. : Addison Wesley, 2000. ISBN: 8478290370.

- **Salavert, Isidro Ramos y Lozano Pérez, María Dolores.** *Ingeniería del software y bases de datos: tendencias actuales.* España : Ediciones de la universidad de Castilla-La Mancha, 2000. ISBN 84-8427-077-7.

- **Sánchez, Lic. Javier Silva.** La autenticación por voz aplicada a la banca telefónica. *Ciencia en su PC.* [En línea] enero de 2010. [Citado el: 15 de octubre de 2013.] http://cienciapc.idict.cu/index.php/cienciapc/article/view/76/236.

- **Tobón, Luis Miguel Echeverry y Carmona, Luz Elena Delgado.** *Caso Práctico de la Metodología Ágil XP al Desarrollo de Software.* Pereira: Universidad Teconológica de Pereira : s.n., 2007.

- **Torres, Gualberto Aguilar, Sánchez Pérez, G. y Toscano Medina, K. .** *Fingerprint Recognition Using Local Features and Hu Moments.* Mexico: Instituto Politécnico Nacional : s.n., 2012.

- UTILIZANDO BIBLIOTECAS DE CÓDIGO C EN APLICACIONES C#. *Revista Telem@tica.* [En línea] [Citado el: 2 de 11 de 2013.] revistatelematica.cujae.edu.cu/index.php/tele/article/download/61/60.

- Visual Paradigm. *Visual Paradigm.* [En línea] [Citado el: 8 de octubre de 2013.] http://www.visual-paradigm.com/.

- **Wilson, Leslie Blacket.** *Comparative Programming Languages .* New York : Addison-Wesley, 1993. ISBN 0-201-56885-3.

Anexos

Anexo 1: Modos de funcionamiento de un SAID.

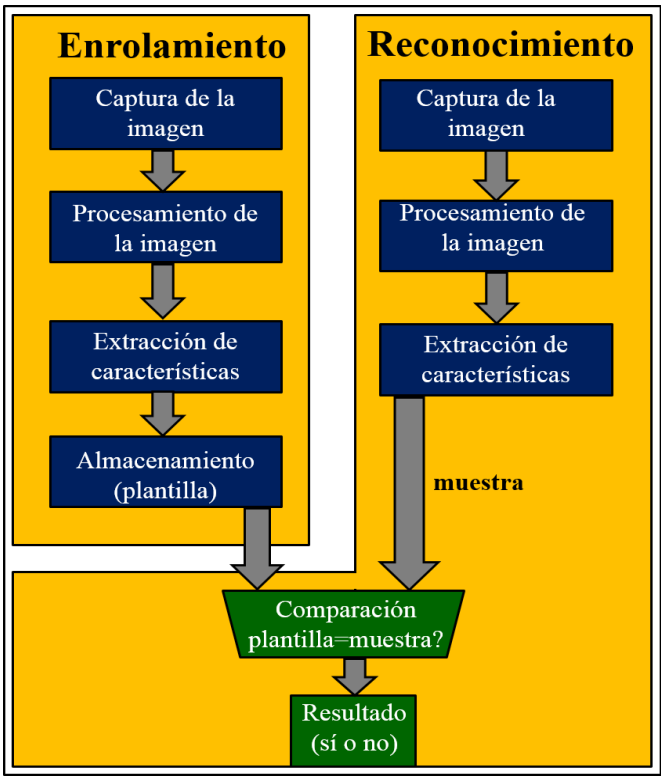


Figura 30 Modos de funcionamiento de un SAID.

Anexo 2: Variación de la condición de la piel.

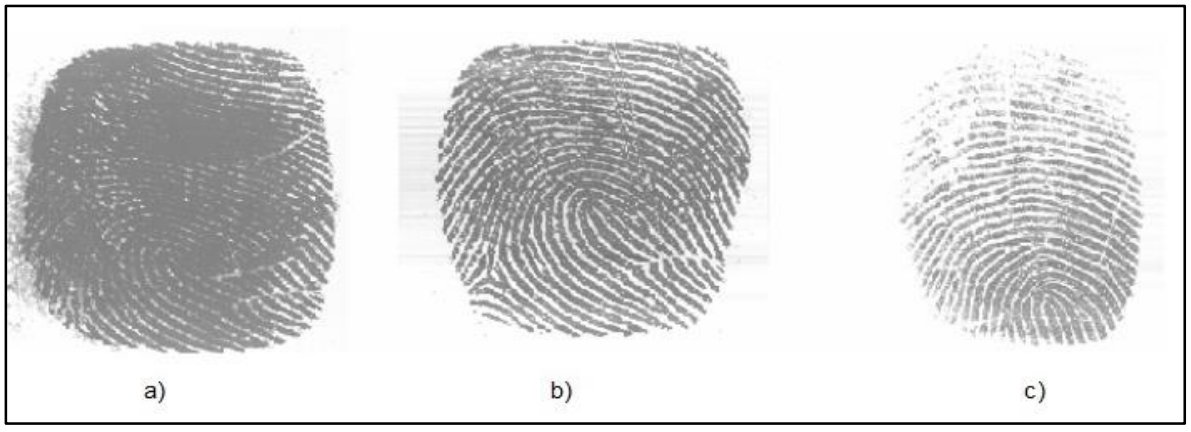


Figura 31 Variación de la condición de la piel: a) piel húmeda b) piel neutral c) piel seca.

Anexo 3: Variación de la condición de la piel.

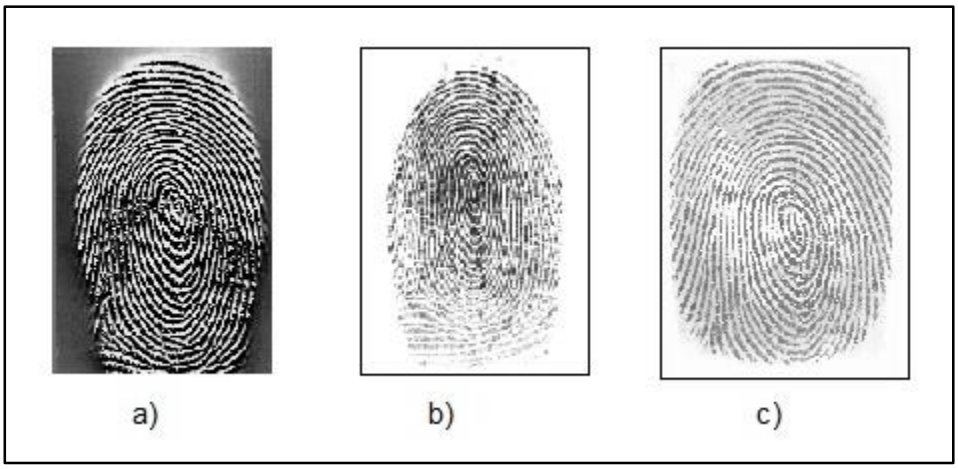


Figura 32 Imágenes de huellas dactilares de un mismo dedo capturadas con distintos sensores: a) sensor térmico b) sensor óptico c) sensor capacitivo.

Anexo 4: Definición del filtro para determinar el contraste direccional utilizado ene l componente desarrollado en la UCI.

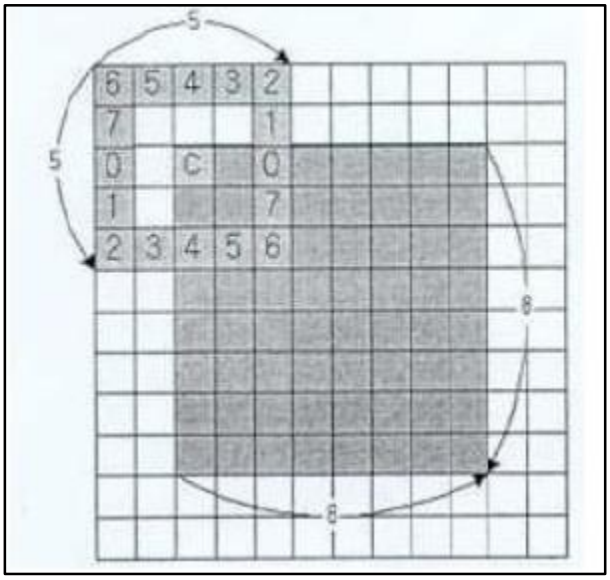


Figura 33 Filtro para determinar el contraste direccional.

Anexo 5: Historias de usuario.

Historia de Usuario	
Número: HU_3	**Usuario:** Sistema
Nombre de historia: Determinar la calidad de un dataset de huellas dactilares.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio

Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Miriela Velázquez Arias	
Descripción: El sistema debe determinar la calidad de un conjunto de imágenes de huellas dactilares ubicadas en una carpeta.	
Observaciones: Se debe haber cargado una carpeta de imágenes de huellas dactilares.	

Tabla 26 HU Determinar la calidad de un dataset de huellas dactilares.

Historia de Usuario	
Número: HU_4	**Usuario:** Miriela Velázquez Arias
Nombre de historia: Generar mapas de calidad de un dataset de huellas dactilares.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 3
Programador responsable: Miriela Velázquez Arias	
Descripción: El componente debe permitir la visualización de las imágenes de huellas de una carpeta. Según se marque una de dichas imágenes, se debe mostrar el mapa de calidad correspondiente a la misma.	
Observaciones: Se debe haber determinado con anterioridad la calidad de las imágenes de huellas dactilares de una carpeta.	

Tabla 27 HU Generar mapas de calidad de un dataset de huellas dactilares.

Anexo 6: Tareas ingenieriles.

Tarea de ingeniería	
Número: TI_1_HU_1	**Historia de usuario:** Determinar la calidad de una imagen de huella dactilar.
Nombre de tarea: Obtener el histograma de intensidades.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 20/01/2014	**Fecha fin:** 24/01/14

Programador responsable: Alexei Alayo Rondón
Descripción: Se debe obtener la frecuencia de aparición de cada nivel de gris de un bloque de la imagen de la huella dactilar, o sea, su histograma.

Tabla 28 TI_HU1 Obtener el histograma de intensidades.

Tarea de ingeniería	
Número: TI_2_HU_1	**Historia de usuario:** Determinar la calidad de una imagen de huella dactilar.
Nombre de tarea: Calcular características locales de un bloque de una imagen.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 27/01/2014	**Fecha fin:** 31/01/2014
Programador responsable: Alexei Alayo Rondón	
Descripción: A partir del histograma de los bloques de una imagen se determina la media, varianza, desviación estándar, suavidad, uniformidad e índice de no homogeneidad de cada bloque.	

Tabla 29 TI2_HU1 Calcular características locales de un bloque de una imagen.

Tarea de ingeniería	
Número: TI_3_HU_1	**Historia de usuario:** Determinar la calidad de una imagen de huella dactilar.
Nombre de tarea: Obtener el espectro de Fourier.	
Tipo: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 3/02/2014	**Fecha fin:** 14/02/2014
Programador responsable: Alexei Alayo Rondón	
Descripción: Se debe obtener el espectro de Fourier a partir de una imagen de huella dactilar.	

Tabla 30 TI3_HU1 Obtener el espectro de Fourier.

Tarea de ingeniería	

Número: TI_4_HU_1	**Historia de usuario:** Determinar la calidad de una imagen de huella dactilar.
Nombre de tarea: Determinar la coherencia local del campo de orientación de los bloques.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 17/02/2014	**Fecha fin:** 21/02/2014
Programador responsable: Alexei Alayo Rondón	
Descripción: Se debe determinar la coherencia del campo direccional de un bloque de la imagen.	

Tabla 31 TI4_HU1 Determinar la coherencia local del campo de orientación de los bloques.

Tarea de ingeniería	
Número: TI_5_HU_1	**Historia de usuario:** Determinar la calidad de una imagen de huella dactilar.
Nombre de tarea: Calificar los bloques de una imagen según las características locales de la misma.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 24/02/2014	**Fecha fin:** 28/02/2014
Programador responsable: Alexei Alayo Rondón	
Descripción: Se debe clasificar los bloques de la imagen en “bueno”, “húmedo” y “seco” a partir de las métricas definidas.	

Tabla 32 TI5_HU1 Calificar los bloques de una imagen según las características locales de la misma.

Tarea de ingeniería	
Número: TI_6_HU_1	**Historia de usuario:** Determinar la calidad de una imagen de huella dactilar.
Nombre de tarea: Determinar el factor de calidad global de una imagen a partir de la concentración de energía en el espectro.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 3/03/2014	**Fecha fin:** 7/03/2014

Programador responsable: Alexei Alayo Rondón
Descripción: A partir del espectro de Fourier de la imagen, calcular un índice de concentración de la energía en el mismo entre las frecuencias 30 y 60.

Tabla 33 TI6_HU1 Determinar el factor de calidad global de una imagen a partir de la concentración de energía en el espectro.

Tarea de ingeniería	
Número: TI_7_HU_1	**Historia de usuario:** Determinar la calidad de una imagen de huella dactilar.
Nombre de tarea: Determinar la calidad general de una imagen de huella dactilar.	
Tipo: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 10/03/2014	**Fecha fin:** 21/03/2014
Programador responsable: Alexei Alayo Rondón	
Descripción: Determinar la calidad general de una imagen de huella dactilar combinando los resultados correspondientes al análisis local y global de la imagen.	

Tabla 34 TI7_HU1 Determinar la calidad general de una imagen de huella dactilar.

Tarea de ingeniería	
Número: TI_1_HU_2	**Historia de usuario:** Generar mapa de calidad de una imagen de huella dactilar.
Nombre de tarea: Obtener el mapa de calidad de una imagen.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 24/03/2014	**Fecha fin:** 28/03/2014
Programador responsable: Alexei Alayo Rondón	
Descripción: Generar gráficamente un mapa de calidad de una imagen de huella dactilar, mostrando zonas específicas de calidad.	

Tabla 35 TI1_HU2 Obtener el mapa de calidad de una imagen.

Tarea de ingeniería	
Número: TI_1_HU_3	**Historia de usuario:** Determinar la calidad de un dataset.

Nombre de tarea: Determinar la calidad de cada una de las imágenes de huellas dactilares de un dataset.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 31/03/2014	**Fecha fin:** 4/04/2014
Programador responsable: Miriela Velázquez Arias	
Descripción: Determinar la calidad de cada una de las imágenes de huellas dactilares de una carpeta, generando un fichero de texto en la propia carpeta con los resultados obtenidos.	

Tabla 36 TI1_HU3 Determinar la calidad de cada una de las imágenes de huellas dactilares de un dataset.

Tarea de ingeniería	
Número: TI_1_HU_4	**Historia de usuario:** Generar mapa de calidad de un dataset.
Nombre de tarea: Obtener el mapa de calidad de las imágenes que conforman un dataset.	
Tipo: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 7/04/2014	**Fecha fin:** 25/04/2014
Programador responsable: Miriela Velázquez Arias	
Descripción: Mostrar en la interfaz todas las imágenes de un dataset de huellas dactilares y, al seleccionar una de ellas, mostrar el mapa de calidad correspondiente.	

Tabla 37 TI1_HU4 Obtener el mapa de calidad de las imágenes que conforman un dataset.

Anexo 7: Tarjetas CRC.

Clase Controler	
Responsabilidades	**Colaboradores**
Aplicar el proceso de medición de calidad y generación del mapa de calidad correspondiente a una imagen de huella dactilar.	• ChaohongWuAlgorithm
Aplicar el proceso de medición de calidad y generación de los mapas de calidad correspondientes a un dataset de huellas dactilares.	• ChaohongWuAlgorithm

Tabla 38 Tarjeta CRC Controler.

Clase BlockProcessing

Responsabilidades	Colaboradores
Determinar la media de un bloque de la imagen.	----------
Determinar la uniformidad de un bloque de la imagen.	----------
Determinar la suavidad de un bloque de la imagen.	----------
Determinar la varianza de un bloque de la imagen.	• MathNet.Iridium
Determinar la desviación estándar de un bloque de la imagen.	• MathNet.Iridium
Determinar la coherencia del campo de orientación de un bloque de la imagen.	----------

Tabla 39 Tarjeta CRC BlockProcessing.

Clase ImageProcessing	
Responsabilidades	**Colaboradores**
Determinar las convoluciones.	• MathNet.Iridium
Obtener la matriz de una imagen.	----------

Tabla 40 Tarjeta CRC ImageProcessing.

Clase TestingInterface	
Responsabilidades	**Colaboradores**
Con el objetivo de verificar el correcto funcionamiento del componente, se crea una interfaz de prueba que permita la visualización del resultado del proceso de medición de calidad, así como los mapas de calidad generados.	----------

Tabla 41 Tarjeta CRC TestingInterface.

Clase ImageBlockQuality	
Responsabilidades	**Colaboradores**
Es un enumerativo que establece las clasificaciones de un bloque de una imagen de huella dactilar.	----------

Tabla 42 Tarjeta CRC ImageBlockQuality.

Clase ImageQuality	
Responsabilidades	**Colaboradores**
Es un enumerativo que establece las clasificaciones de calidad de una imagen de huella dactilar en.	----------

Tabla 43 Tarjeta CRC ImageQuality.

Clase ImageQualityLevel	
Responsabilidades	**Colaboradores**
Es un enumerativo que establece los niveles de calidad de una imagen de huella dactilar.	----------

Tabla 44 Tarjeta CRC ImageQualityLevel.

Anexo 8: Código del método encargado de la medición de calidad de imágenes de huellas dactilares.

```
public int FingerprintQuality(Bitmap image)
        {
            _globalAnalisys = new GlobalQualityAnalysis(image);
            _localAnalisys = new LocalQualityAnalysis(image);

            QualityEstimationMap localQuality = _localAnalisys.LocalQuality();
            double globalQuality = _globalAnalisys.GlobalQuality();

            int blocksAmount = localQuality.Block.GetLength(0) * localQuality.Block.GetLength(1);

            float goodBlockPercentage = 0, normalBlockPercentage = 0, wetBlockPercentage = 0,
dryBlockPercentage = 0, corruptedBlockPercentage = 0, canNotDetermineBlockPercentage = 0;
            int goodBlockCount = 0, normalBlockCount = 0, wetBlockCount = 0, dryBlockCount = 0,
corruptedBlockCount = 0, backgroundBlockCount = 0, canNotDetermineBlockCount = 0;
            int goodCount = 0, normalCount = 0, wetCount = 0, dryCount = 0, spoiledCount = 0;

            for (int i = 0; i < localQuality.Block.GetLength(0); i++)
                for (int j = 0; j < localQuality.Block.GetLength(1); j++)
                    switch (localQuality.Block[i, j])
                    {
                        case BlockQuality.good:
                            goodBlockCount++;
                            break;
                        case BlockQuality.normal:
                            normalBlockCount++;
                            break;
                        case BlockQuality.wet:
                            wetBlockCount++;
                            break;
                        case BlockQuality.dry:
                            dryBlockCount++;
                            break;
                        case BlockQuality.corrupted:
                            corruptedBlockCount++;
```

```
                            break;
                        case BlockQuality.background:
                            backgroundBlockCount++;
                            break;
                        default:
                            canNotDetermineBlockCount++;
                            break;
                    }

            goodBlockPercentage = (float)((float)goodBlockCount / (float)(blocksAmount -
backgroundBlockCount)) * 100;
            normalBlockPercentage = (float)((float)normalBlockCount / (float)(blocksAmount -
backgroundBlockCount)) * 100;
            wetBlockPercentage = (float)((float)wetBlockCount / (float)(blocksAmount -
backgroundBlockCount)) * 100;
            dryBlockPercentage = (float)((float)dryBlockCount / (float)(blocksAmount -
backgroundBlockCount)) * 100;
            corruptedBlockPercentage = (float)((float)corruptedBlockCount / (float)(blocksAmount -
backgroundBlockCount)) * 100;

            canNotDetermineBlockPercentage = (float)((float)canNotDetermineBlockCount /
(float)(blocksAmount - backgroundBlockCount)) * 100;

            if (goodBlockPercentage + normalBlockPercentage >= 70)
            {
                if (goodBlockPercentage > normalBlockPercentage)
                    goodCount++;
                else
                    normalCount++;
            }

            if (normalBlockPercentage + dryBlockPercentage >= 70)
            {
                if (normalBlockPercentage > dryBlockPercentage)
                    normalCount++;
                else
                    dryCount++;
            }

            if (normalBlockPercentage + wetBlockPercentage >= 70)
            {
                if (normalBlockPercentage > wetBlockPercentage)
                    normalCount++;
                else
                    wetCount++;
            }

            if (dryBlockPercentage + wetBlockPercentage + corruptedBlockPercentage >
normalBlockPercentage + goodBlockPercentage)
            {
                if (corruptedBlockPercentage > dryBlockPercentage && corruptedBlockPercentage >
wetBlockPercentage)
                    spoiledCount++;

                else if (dryBlockPercentage > corruptedBlockPercentage && dryBlockPercentage >
wetBlockPercentage)
                    dryCount++;

                else if (wetBlockPercentage > corruptedBlockPercentage && wetBlockPercentage >
dryBlockPercentage)
```

```
                    wetCount++;
            }

            if (goodBlockPercentage >= 70 && globalQuality >= 4.90)
                goodCount++;

            else if (goodBlockPercentage < 70 && globalQuality >= 4.90)
            {
                if (goodBlockPercentage >= 50 && goodBlockPercentage > wetBlockPercentage &&
goodBlockPercentage > dryBlockPercentage)
                    normalCount++;

                else if (wetBlockPercentage >= 30 && globalQuality > 5.20)
                    wetCount++;

                else if (dryBlockPercentage >= 30 && globalQuality <= 5.20)
                    dryCount++;

                else if (goodBlockPercentage <= 30 && corruptedBlockPercentage >= 30)
                    spoiledCount++;
            }

            else if (goodBlockPercentage >= 70 && globalQuality < 4.90)
            {
                if (globalQuality >= 4.0)
                    normalCount++;
                else
                    dryCount++;
            }

            if (goodBlockPercentage >= 50 && goodBlockPercentage < 70 && globalQuality >= 4.0 &&
globalQuality < 4.90)
                normalCount++;

            else if (goodBlockPercentage < 50 && globalQuality >= 4.0 && globalQuality < 4.90)
            {
                if (goodBlockPercentage >= 30)
                    if (dryBlockPercentage > 30)
                        dryCount++;

                    else if (corruptedBlockPercentage > 30)
                        spoiledCount++;

                else if (goodBlockPercentage < 30)
                {
                    if (dryBlockPercentage >= 30)
                        dryCount++;

                    else spoiledCount++;
                }
            }

            else if (goodBlockPercentage >= 50 && goodBlockPercentage < 70 && globalQuality < 4.0)
                dryCount++;

            else if (goodBlockPercentage < 50 && globalQuality < 4.0)
               dryCount++;

            if (globalQuality < 4.0 && goodBlockPercentage <= 30 && wetBlockPercentage >= 30)
                wetCount++;
```

```
            if (globalQuality <= 4.0 && goodBlockPercentage <= 30 && dryBlockPercentage >= 30)
                dryCount++;

            if (corruptedBlockPercentage >= 30)
                spoiledCount++;

            int[] clasificaciones = { goodCount, normalCount, wetCount, dryCount, spoiledCount };

            int mayor = 0;
            for (int i = 0; i < clasificaciones.Length; i++)
            {
                if (clasificaciones[i] > clasificaciones[mayor])
                    mayor = i;
            }

            return mayor + 1;
        }
```

Anexo 9: Casos de prueba de aceptación.

Caso de Prueba de Aceptación	
Código: CP3_HU3	**HU_1:** Determinar la calidad de un dataset.
Responsable: Miriela Velázquez Arias	
Descripción: Prueba de funcionalidad para verificar la medición de calidad de una carpeta de huellas dactilares.	
Condiciones de Ejecución: El proceso debe recibir una carpeta de huellas dactilares.	
Entrada / Pasos de ejecución: Luego de cargar el dataset de huellas dactilares en la interfaz de prueba, el sistema debe determinar la calidad de cada una de las huellas dactilares que lo conforman.	
Resultado Esperado: Se genera un fichero de texto en la propia carpeta con el nombre "**DatasetQuality**" con los valores de calidad asociados a cada imagen de huella dactilar del dataset.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 45 Caso de prueba de aceptación 3.

Caso de Prueba de Aceptación	
Código: CP4_HU4	**HU_1:** Generar el mapa de calidad de un dataset de huellas dactilares.
Responsable: Miriela Velázquez Arias	

Descripción: Prueba de funcionalidad para verificar la visualización del mapa de calidad de cada huella de un dataset.
Condiciones de Ejecución: El proceso debe recibir una carpeta de huellas dactilares.
Entrada / Pasos de ejecución: Luego de cargar un dataset de huellas dactilares en la interfaz de prueba, el sistema genera los mapas de calidad asociados a cada una de las imágenes de huellas dactilares del dataset.
Resultado Esperado: Se genera una carpeta dentro del dataset con el nombre "**Results**" que contiene los mapas de calidad asociados a cada imagen de huella dactilar del dataset.
Evaluación de la Prueba: Prueba satisfactoria.

Tabla 46 Caso de prueba de aceptación 4.

Anexo 10: Interfaz de prueba.

Figura 34 Interfaz que muestra el procesamiento de un dataset de huellas dactilares.